Active Shape Completion Using Tactile Glances

Jonas Jung

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 25.11.2019

Supervisor

Prof. Ville Kyrki

Advisors

Dr. Francesco Verdoja

M.Sc. Jens Lundell

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Jonas Jung

Title Active Shape Completion Using Tactile Glances

Degree programme Erasmus Mundus Space Master

Major Space Robotics and Automation **Code of major** ELEC3047

Supervisor Prof. Ville Kyrki

Advisors Dr. Francesco Verdoja, M.Sc. Jens Lundell

Date 25.11.2019 **Number of pages** 46+16 **Language** English

Abstract

One longstanding challenge in the field of robotics has been the robust and reliable grasping of objects of unknown shape. Part of that challenge lies in reconstructing the object's shape using only limited observations. Most approaches use either visual or tactile information to reconstruct the shape, having to face issues resulting from the limitations of the chosen modality. This thesis tries to combine the strengths of visual and tactile observations by taking the result from an existing visual approach and refining that result through sparse tactile glances.

The existing approach produces potential shape hypotheses in voxel space which get combined into one final shape. This thesis takes that final shape and determines voxels of interest using either entropy or variance. These voxels will be targeted by the exploration, providing information about these voxels. This information will be used to assign weights to the original hypotheses in order for the combined shape to better fit the observations.

All explorations are simulated and evaluated in MATLAB. The resulting shapes are evaluated based on their Jaccard Index with the ground truth model. The algorithm leads to improvements in the Jaccard Index, but not to drastically different looking shapes.

Keywords shape completion, haptic exploration, robotics

Contents

Symbols and abbreviations

Symbols

$\vec{a}$	Approach vector
B_i	Value of histogramm bin i for one voxel
E	Expected value
f	Potential function for shape representation
f_*	Gaussian Process posterior
H	Entropy of one voxel
H_{all}	Entropy of all voxels
k	Kernel function
K	Covariance matrix
L	Location of Voxel in voxel space (x, y, z) with $x, y, z \in \mathbb{N}_{<=40}$
N	Number of hypotheses
$\mathbf{N}$	Neighborhood voxels
occ	Value of voxel in ground truth, 1 = occupied, 0 = empty
p	Probability function
$\mathbf{P}$	Pointcloud, list of points in three-dimensional space $p_i = (x, y, z)$
res	Value of voxel in weighted mean shape after thresholding
$\mathbf{S}$	Set of all hypotheses
S_i	Value of voxel in hypothesis i
$\mathbf{T}_i$	Partial truth voxels after i iterations
v	Voxel
v_c	Contact voxel
v_t	Target voxel
$\mathbf{V}_{path}$	Path voxels
$\mathbf{X}$	Random variables
δ	Stepsize parameter
μ	Mean
μ_ω	Value of voxel in weighted mean shape
σ^2	Variance
σ_ω^2	Weighted variance of one voxel
ω_i	Weight of hypothesis S_i

Abbreviations

GP	Gaussian Process
MLS	Moving Least Squares
NN	Neural Network
SQP	Sequential Quadratic Programming

1 Introduction

Robots have become increasingly relevant in our lives, already finding use in countless applications. Ranging from industrial assembly arms to automated vacuum cleaners and self-driving cars, robots come in all shapes and sizes. However, upon hearing the word "robot" most people instinctively picture humanoid robots that resemble a person, can perform daily tasks and might even possess some form of intelligence that allows them to speak. These humanoid robots, or androids, are still in the distant future, with innumerable problems needing to be overcome before androids become a reality. One of these problems is the seemingly simple task of grasping an object of unknown shape.

While humans subconsciously perform this task on a daily basis, a robot is faced with a surprising number of variables that make this task very challenging. These include the object's shape, compliance, total weight and weight distribution, the distance to the object, the friction of different contact points and the constraints on how the object is allowed to be moved. Humans are able to combine a wide array of sensor signals with years of experience to overcome this challenge, while no robot has yet been able to reliably do so. Finding a robust solution would allow for much more flexible robots not limited to a single task. Possible applications include medical or personal assistance robots that could tend to a person's needs in an unknown environment.

The challenge lies not only in combining all of the above-mentioned variables but also in obtaining their true value. To obtain the object's shape, measurements can be made visually [1] using cameras and depth sensors or haptically [2] using force sensors. Although visual measurements have the advantage of being dense and accurate, they are limited to the camera-facing side of the object, meaning that nothing is known about occluded surfaces. Conversely, haptic measurements are not limited to one side and can evenly explore all surfaces but are less dense and require more time than visual measurements.

Both approaches face the issue of incomplete information. Measurements will only cover sections of the objects, leaving unknown regions that need to be reconstructed. The problem of filling in the missing information and modelling the entire surface from incomplete observations is referred to as shape completion. Possible approaches for shape completion include Neural Networks (NN) [3] and Gaussian Processes (GP) [4]. One specific NN approach is to train a NN to convert a single depth image of an object into multiple possible shapes (i.e., hypotheses) [5]. These hypotheses are then combined into one mean shape, representing the best fit from the observed data.

However, these approaches only consider one type of measurement (visual or haptic), leaving them with the issues mentioned earlier. Attempts have been made to combine visual and tactile measurements in order to take advantage of their respective strengths [6]. For this purpose, a visual measurement has been used to generate an initial estimate of the shape, thus providing accurate information about half of the object. Afterwards, tactile measurements can be performed in occluded regions to obtain information about uncertain surfaces and to refine the initial estimate into a more accurate reconstruction. However, while in [6] haptic and

visual measurements get combined successfully, they use a very dense and expensive touch sensor which assumes many touches. The assumption of this thesis is that full hypotheses generated only from visual information can be effectively combined using only sparse and inexpensive haptic exploration.

Therefore, the aim of this thesis is to develop an approach that uses haptic exploration for refining the weights of the hypotheses generated by the NN developed in [5] in order for the mean shape to more accurately resemble the real object. Some areas of the shape carry more information than others because of disagreement between the hypotheses concerning unobserved surfaces. To quantify the amount of information in an area, multiple metrics will be introduced to determine the points of interest that need to be touched in order to gain meaningful information about the object's shape. The touch will then be executed in an idealized, simulated environment, allowing the contact location, together with the path of empty space taken by the fingertip to be recorded. These locations, confirmed as empty or filled, will then be used as a partial ground truth to calculate the new weights of the hypotheses, resulting in a new combined shape. The weights will be chosen to maximize the overlap between the resulting shape and the partial ground truth.

The resulting shape will be evaluated based on its Jaccard index with the real shape. Using the Jaccard Index, different parameters and design choices will be compared to each other and evaluated.

The thesis is divided into the following chapters: Chapter 2 serves as the literature review, explaining possible methods for shape completion and reviewing previous research on the different approaches. Chapter 3 explains the method used to refine the weights, which is divided into determining the contact location and using the contact location to update the weights. Chapter 4 describes the implementation of the simulations that serve as the experiments to evaluate the effectiveness of the algorithm. Chapter 5 will summarize the results obtained from the simulations. Chapter 6 concludes by discussing the results and suggesting possible future improvements.

2 Shape Completion

This chapter will provide the relevant background to the problem of shape completion. The first section will introduce and explain established models for representing shapes, highlighting the respective strengths and weaknesses. The second section reviews past research on shape completion, more specifically shape completion with either visual or tactile observations, the two types of measurements relevant for the approach presented in this thesis.

2.1 Models for representing shapes

This section presents different models used to represent three dimensional shapes. The three presented models are mesh representation, Gaussian Process (GP) and voxelization. Others, such as superquadratics [7], exist, but will not be explained here, since all the reviewed past research in the next section use one of the three aforementioned models.

2.1.1 Meshes

The most common model to represent arbitrary three-dimensional shapes is through polygonal meshes. These are a common structure in computer graphics and are defined through vertices and faces [8]. The vertices make up the edges of the shape and are defined by their location in three-dimensional space. The faces are defined as a subset of vertices that together form a surface of the shape. Most common are triangular meshes where each face consists of three vertices, resulting in triangles. Figure 1 shows an example of a triangulated mesh of a rabbit.

The big advantage of mesh representations is their ability to cope with uneven sampling, allowing for a high resolution, only limited by the sampling density [9]. This ability to place vertices in arbitrary locations from uneven sampling, allows to efficiently store high resolution features with larger, low resolution features. Every

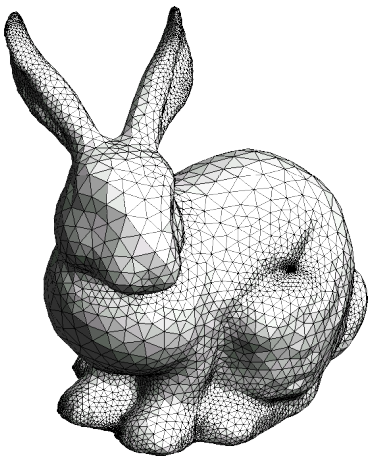


Figure 1: Example triangular mesh of a rabbit.

vertex and every face requires the same amount of memory, regardless of location and size, so larger areas can be compressed through larger faces.

2.1.2 Gaussian Process representation

An object can be thought of as a function in three-dimensional space where the value of the function describes whether a point in space is occupied by the object or not. Such a representation is called an implicit surface and is defined as the zeroes of a potential function f with

$$f : \mathbb{R}^d \longrightarrow \mathbb{R}; f(x) \begin{cases} = 0, & \text{x lies on the surface} \\ > 0, & \text{x lies outside the object .[4]} \\ < 0, & \text{x lies inside the object} \end{cases} \tag{1}$$

The challenge lies in finding a function f that adequately describes the shape of the object. A solution is offered by the Gaussian Process (GP) by assuming an infinite distribution of functions and using observations to refine that distribution into the most fitting function [10].

The GP can be viewed as an infinite dimensional extension of a multivariate Gaussian distribution, which, in turn, presents a generalization of the one-dimensional Gaussian or normal distribution. A normal distribution describes the likelihood for all possible values of a random variable X with the probability density function

$$p(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

where $p(x)$ is the probability of X having the value x, μ is the mean and the most likely value of the distribution and σ^2 the variance which describes how far the variable usually deviates from the mean. This distribution looks like a bell with the mean being the most likely value. Theoretically no value is impossible, but the probabilities are decreasing exponentially with increasing distance from the mean. Many random processes produce normal distributions, one important example being some measurement errors. That is the reason why noisy, unbiased measurements can be repeated to have their mean converge to the real value.

This distribution can be generalized to multiple, correlated variables. Given a k-dimensional vector of random variables $\mathbf{X} = (X_1, ..., X_k)$ a multivariate distribution is uniquely defined by a mean vector $\mu = E[\mathbf{X}] = (E[X_1], ...E[X_k])$ and a $k \times k$ covariance matrix $\sum_{i,j} := E[(X_i - \mu_i)(X_j - \mu_j)]$. The mean vector μ and the diagonal covariance matrix elements $\sum_{i,j}$ where $i = j$ can be interpreted like the mean μ and the variance σ^2 from the one-dimensional case. The remaining covariance elements $\sum_{i,j}$ where $i \neq j$ denote the correlation between the different variables. A positive correlation between two variables means if one variable deviates from the mean, the other one is likely to deviate in the same direction and vice versa. A negative correlation means the variable will mirror the deviations in the opposite directions and no correlation means the two variables are independent of each other. The correlation can therefore be interpreted as a measure of similarity.

The GP models a function f(x) as an infinitely-dimensional multivariate distribution with the range X as possible variables. The idea is that some points of the function are known and the missing ones can be interpolated from the given ones, similar to regression. Observed values of f are treated as samples drawn from a multivariate distribution with all observed locations X forming the input-space. The GP approach assumes a smooth function, meaning that the function f(x) has similar values for similar x. That concept of similarity between neighbouring points is reflected in a high covariance between similar inputs x. The key characteristic that practically defines a GP in this application is the way to measure similarity of the inputs. That is done using a covariance or kernel function, which differs from the covariance matrix used in a multivariate distribution by being continuous. The multivariate distribution only has a finite set of variables, meaning a matrix is sufficient to assign a covariance to every pair of inputs. The Gaussian process however has an infinite number of continuous inputs, requiring a function that defines the covariance for any pair of inputs. One example of a commonly used covariance function is the squared exponential kernel

$$k(x_i, x_j) = \sigma^2 e^{-\frac{(x_i - x_j)^2}{2l^2}} \tag{3}$$

with x_i and x_j being two inputs, σ^2 being the signal variance and l being a tuning parameter to determine how fast similarity should decrease with increasing distance.

To obtain a function through a GP, the GP has to be trained with values from that function. Assume $f(\mathbf{x})$ is the set of given values for the corresponding locations $\mathbf{x}$. The kernel function $k(x_i, x_j)$ can then be used to calculate a covariance matrix $K(x, x')$ for all pairs of locations in $\mathbf{x}$. That covariance matrix can then be used to draw samples at unknown locations $\mathbf{x}_*$, because the kernel function allows us to get the covariance $K(x_*, x)$between the unknown locations and the known ones. Since the new values are from the same distribution as the known ones and we know the correlation between the known and the unknown values, we can calculate the mean and the variance for the yet unknown ones. The expected mean value is

$$f_*(\mathbf{x}_*) = K(\mathbf{x}_*, \mathbf{x})K(x, x')^{-1} f(\mathbf{x}) \tag{4}$$

which can be regarded as the best fit function for the observed samples $f(\mathbf{x})$.

To summarize, all a GP needs to uniquely define a function is a kernel function to measure similarity and a set of known samples from that function. The more samples are known, the better the reconstruction of the underlying function. However, the computational cost is proportional to the cube of the number of samples, meaning a GP may become unfeasible if too many samples are required to adequately reconstruct a shape.

2.1.3 Representing a shape in voxel space

An intuitive way to represent and visualize shapes is through voxelization. A voxel is a datapoint in an evenly spaced three-dimensional grid, similar to pixels in two dimensions [11]. Any shape representation can be converted into voxelspace by

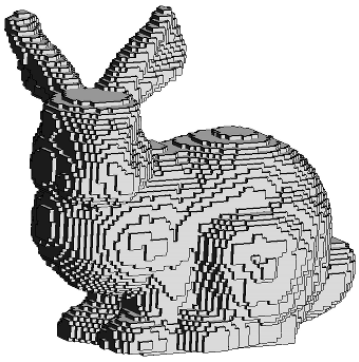


Figure 2: Voxelized rabbit. Source: Bhowmick, 2011 [12]

checking for each voxel if that space is occupied by the shape or not. Figure 2 shows the same rabbit as Figure 1 but voxelized instead of triangulated.

A voxelized shape is usually defined through the list of occupied voxels. Since the number of possible voxels grows cubically with the resolution, the required storage space can quickly exceed feasibility if small details are required. It is therefore recommended to use voxelized shape when the general features are more important than fine details.

2.2 Previous research

A lot of work has already been done in the field of shape completion for robotics. The two approaches related to this work are shape completion based on vision and based on tactile information. This section will review previous approaches that made use of one type of sensor information as well as approaches that combined vision and tactile observations.

2.2.1 Vision based shape completion

Modern cameras and scanners are able to provide high-resolution depth images of objects. These depth images can be transformed into dense and precise point clouds $\mathbf{P}$, which are a list of points $p_i = (x, y, z)$ on the surface observed by the depth-sensor. Unfortunately, these point clouds are incomplete since a depth image will only capture the camera-facing surfaces, with the back side being occluded. Several approaches have been studied to reconstruct the full shape from partial point clouds. Figure 3 shows an example object and the point cloud observed by a depth sensor.

Surface reconstruction

Traditionally the problem of shape completion was concerned about recovering smooth, airtight surfaces from point clouds [13], without considering large occluded or missing regions in the data. No matter how complete or how dense, a point cloud

Figure 3: Bottle and observed point cloud.

will always be a collection of points and not a complete surface, which is why surface reconstruction methods are required.

Maybe the very first method was the conversion of a point cloud into a signed distance field by Hoppe et al. in 1992 [14]. First, they approximate a tangent plane for each datapoint p_i, resulting in an oriented surface normal n_i for each point. This allows the calculation of the distance from each point in space x to the tangent plane of the closest point in P, resulting in the signed distance function $f(x)$:

$$f(x) = (x - p_i) \cdot n_i \tag{5}$$

The surface is defined as the zero level of the signed distance function. While easy to understand and calculate, this method is very sensitive to artefacts in the point cloud, such as noise or non-uniform sampling, not to mention missing regions of the point cloud.

To address the issue of noise sensitivity, one approach is the employment of partial polynomial fits instead of a signed distance function [15]. This approach is called the Moving Least Squares (MLS) approach, because it fits the polynomials by minimizing the squared error between the function and the data points and then moves the polynomial over the entire surface, refitting it at every point. Many implementations of the MLS approach have been studied [16] [17] [18] [19], which use different functions representing the surface, different weighting function determining the influence of neighbouring points and different assumptions like the uniformity of the samples or the orientation of normals.

They all share the strength of noise robustness due to including multiple points in the estimation of each partial fit, enabling the majority of the points to compensate for outliers or noise in the data. Most implementations even include parameters in their

weighting function, allowing the user to fine tune the influence of neighbouring points, to compensate for varying degrees of noise. Furthermore, nonuniform sampling can be addressed with dynamic weighting functions to increase the influence range in sparsely sampled regions. However, since these methods fit locally, their performance still drops drastically in regions with missing data.

Geometry based approaches

One approach to fill in missing regions is to predict and exploit symmetries of common household objects. The partial point cloud gets scanned for planar [20] or rotational symmetries [21] and the occluded regions get reconstructed according to the most fitting symmetry found. This approach can only be successful if the object has symmetries to exploit. While that might be true for many common household objects, more complex shapes require a different solution.

Recognition approach

Another popular approach could be summarized as the recognition approach, which tries to recognize the partial shape in a given set of pre-modelled, complete shapes [22]. The recognition is achieved by matching shape descriptors from the partial shape to the dataset, one commonly used descriptor being features generated from point pairs [23] [24] [25]. Because of the accuracy of the given partial point cloud, shapes can be matched reliably even when introducing clutter and additional occlusions [26]. However, the significant drawback of this approach lies in its dependency on pre-existing models. Creating accurate ground truths requires significant effort, quickly becoming unfeasible for real world environments. Although attempts at object recognition with more easily creatable ground truth models have been successful [27], compiling a complete dataset will never become feasible.

Learning approach

More recently artificial neural networks (NN) have been adopted to reconstruct the complete shape [1] [3]. A network gets trained by feeding it partial and complete point cloud pairs. These pairs offer the NN a way to extract features from a partial point cloud and map them to features in complete shapes. After offline training is completed, the NN is then able to extract features from novel point clouds and predict the features of the complete shape. That approach is similar to the recognition approach in their reliance on pre-modelled ground truths. The significant difference is that the NN is able to interpolate from the trained models, removing the requirement of an exhaustive ground truth dataset. While state of the art implementations have shown impressive reconstructions [28], these reconstructions will always be limited by the features in the training data and the information provided by the depth image.

2.2.2 Tactile based shape completion

Instead of observing an object, information about its shape can be gained through haptic exploration. Tactile sensors might not be able to scan entire surfaces instantaneously like optical sensors, but they have the advantage of reaching occluded regions and exploring all surfaces equally.

Sommer et al. have successfully shown how a robot can explore an object's surface with both hands simultaneously [29]. The tactile sensors in the robotic hands record a point cloud representing the object's surface and that pointcloud can then be matched to previously taught point clouds. The approach is similar to the recognition approach described for visual information, with the partial point cloud being recorded through tactile sensors instead. The limitation remains, that unless the entire object gets probed, a dataset of pre-recorded shapes is required.

Many studies model the shape using a Gaussian Process (GP). GPs present a powerful tool to interpolate values in between observations, complementing the uniform exploration enabled by tactile sensors [4]. Some approaches have been to procedurally touch the surface point by point [2] or slide the robotic hand across the surface [30] to record more data at the same time. The robot was able to autonomously navigate the hand to regions of high uncertainty based on the inherent variance metric of a GP. The exploration will conclude once the variance falls below a threshold, meaning the model is confident enough about the constructed shape. The issue with purely haptic exploration is the considerable time required to extensively explore the entire object.

2.2.3 Multi-modal sensor fusion

A promising approach to deal with the flaws of unimodal shape completion is the combination of visual and tactile information. Recent research has tried to use sparse tactile information to refine an initial shape estimate obtained through visual information. One possible implementation is by modelling the shape with a GP and using a depth image to obtain an initial hypothesis [6]. That initial estimate can then by refined by touching the object in high variance locations and updating the GP with the new observations. That way considerably less touches are required to obtain an accurate shape compared to having no vision-based initial estimate. The slight drawback of using GP in this instance is that the predicted backside of the object will be really unreliable before some touches have occurred.

To get better initial estimates, recent works have used NNs instead of GPs to complete occluded regions. Watkins-Valls and colleagues have expanded upon their own work on using a NN to complete incomplete point clouds by successfully incorporating tactile data into the input of the NN [31]. They augment their point cloud generated from a depth-image by a point cloud generated through random touches on the occluded surfaces and are therefore able to provide the NN with information about the backside of the object. The result is a significantly more accurate prediction about the occluded regions.

Instead of touching the object randomly, studies have been successful in proposing metrics to choose where to touch the object in order to gain more information than

by touching it randomly . Wang et al. [32] represent the model in voxel space, allowing them to target regions where the voxel values are close to 0.5, representing uncertain voxels. They then use the gained information about all the voxels that were passed by the end-effector to calculate a loss function. By backpropagating the gradient of that loss function through the NN, they can adjust the prediction to match the observed voxels. However, they use a very rich depth-sensor enabling very dense sampling of the object's surface, resulting in plenty of data to backpropagate. In contrast, this thesis tries to refine the estimate with significantly fewer and sparser haptic measurements.

A different method is presented in [33] that generates tactile data by grasping the model generated from a depth sensor image and symmetry assumptions. The fusion of the visual and tactile data is formulated as a state estimation problem and solved with an iterative extended Kalman filter. They show an improvement of shape reconstruction accuracy compared to using only one type of sensor information but use symmetry assumptions in both the initial model as well as the refined model. The method in this thesis does not assume symmetries, allowing for a wider range of objects to be reconstructed.

2.2.4 Uncertain shape completion as prior

This thesis will be a direct extension of the uncertain shape completion in Lundell et al. [5]. They trained a deep neural network to take a single depth image of an object as input and outputs a user-defined number of hypotheses shapes in voxel space, that combine into the completed shape. It therefore fits into the learning approach category with vision-based observations.

The neural network used follows the architecture proposed in [34] with the addition of dropout layers [35] to prevent overfitting and to enable them to estimate the network uncertainty following a procedure called Monte-Carlo Dropout [36]. Since the output is in voxel space, combining the hypotheses into one mean shape is straight forward through

$$\mu = \frac{1}{N} \sum_{i=1}^{N} S_i \tag{6}$$

with μ being the mean shape, and S_i being one of the N hypotheses.

The goal of this thesis is to develop an algorithm that improves the mean shape, with the help of tactile glances as new observations. The hypotheses will be used to determine voxels of interest that will be targeted by haptic exploration to gain new information about the shape. This new information will then be used to assign weights to the hypotheses, resulting in a novel mean shape that hopefully better fits all observations.

3 Active Haptic Exploration

This chapter will outline the exact problem that needs to be solved before explaining the method used to overcome the problem. The proposed solution will be an algorithm that refines the weights from the hypotheses generated by [5] in order to generate a better fitting shape estimate. The algorithm can be divided into three steps: Selecting a target voxel, approaching that voxel until contact occurs, and updating the weights based on the contact location.

3.1 Problem statement and methodology

The problem that needs to be solved is to generate and incorporate tactile measurements of a partially known shape with shape hypotheses generated by [5] in order to produce a more complete and better fitting shape estimate. The sub-problem of generating informative tactile measurements consists of using the existing shape estimates (hypotheses) to automatically identify regions of interest, where tactile measurements will generate useful, new information about the shape. The other sub-problem, the incorporation of the tactile measurements, deals with the question of how the new data can be used to combine the hypotheses into a more accurate shape. For that purpose weights ω_i will be introduced, that determine the influence of each hypothesis when being combined into the mean shape. Instead of combining the hypotheses according to Equation 6, the hypothesis will be multiplied by their respective weight, resulting in

$$\mu_\omega(v) = \frac{1}{\sum_{i=1}^{N} \omega_i} \sum_{i=1}^{N} \omega_i \cdot S_i(v). \tag{7}$$

These weights allow the hypotheses to combine into novel shapes. By basing these weights on the haptic exploration it becomes possible to change the final shape estimate according to the new tactile information, while maintaining dependencies generated by the NN.

Voxel space

The hypotheses generated by [5] are represented in voxel space, and since voxels are easily visualized and manipulated, the entire thesis will operate in voxel space. A voxel v in this thesis has the following properties:

- Location in voxel-space $L(v) = (x, y, z)$ with $x, y, z \in \mathbb{N}_{\leq 40}$
- True value in the ground truth $occ(v) = \begin{cases} 0, & \textit{voxel is empty} \\ 1, & \textit{voxel is occupied} \end{cases}$
- One value for each hypothesis $S_i(v) = [0, 1]$
- The resulting value after combining each of the N hypotheses into a mean shape $\mu_\omega(v)$ (see Equation 7)
- The resulting value after thresholding the mean shape $res(v) \in \{0, 1\} = \lfloor \mu_\omega(v) \rceil$

Algorithm

The proposed solution to the problem mentioned above can be divided into three steps:

- Locating the voxel with the highest expected information gain when contacted.
- Approaching that voxel and recording the contact location.
- Using the contact location to assign new weights in order for the final shape estimate to better resemble the new observations.

3.2 Selecting target voxel

The first step is identifying a voxel of interest, or target voxel v_t, that can be touched to gain meaningful information about the real shape. To quantify how interesting each voxel is, a metric is required that provides each voxel a measure of the expected informativeness of touching it and that can be calculated from the weights and the hypotheses. Three different metrics will be presented where the target voxel will be the one with the highest value assigned by that metric. The three metrics will be evaluated against each other and against random exploration paths as a baseline. The random exploration will pick a random starting point on the voxel grid border and a random direction. If this path leads to no contact it will be disregarded and a new random path will be generated until a contact occurs. The other three approaches have no guarantee to get a contact. Nevertheless, missing occasionally is acceptable, since misses give information as well. Missing too often however, will decrease the performance drastically, which is the reason the random approach has a built-in contact guarantee. Without it, the approach would miss more often than succeed.

3.2.1 Reducing maximum local variance

The first metric is the variance of the hypotheses in each voxel. The variance is a measure of how much the individual observations differ from their mean value. In other words, the variance approaches zero when all observations become more similar to each other and the variance peaks when the observations are split evenly at the borders of their range. In this specific case, the observations are the values of the different hypotheses for the same voxel. If all of the hypotheses are weighted equally, which is the case before the first touch, the variance σ^2 of one voxel v can be calculated by

$$\sigma^2(v) = \frac{1}{N} \sum_{i=1}^{N} |S_i(v) - \mu(v)|^2 \tag{8}$$

where N is the number of hypotheses, $S_i(v)$ the value of hypothesis S_i at voxel v and $\mu(v)$ the mean value of that voxel, calculated by Equation 6. This variance can already be used as a metric to determine a voxel of interest for the first touch. Voxels where a touch would be wasted are voxels with low variance. That is because a low variance means that all of the hypotheses already agree about that voxel. Not only

does this mean that the resulting value of that voxel cannot be changed by changing the weights, it also means that the model is already certain about that voxel and a change would be unnecessary. Most of these low variance, already certain voxels will be voxels that got directly observed by the depth sensor used in [5] to generate the hypotheses. Touching them would only tell the model what it already knows.

Conversely, voxels with a high variance are the exact opposite and the recommended locations for a contact. They represent locations where the hypotheses disagree about the value, meaning the model is still uncertain about that voxel. Determining the real value of that high variance voxel through a contact would enable the model to disregard the conflicting hypotheses and assign the agreeing ones more weight. These high variance voxels will be the ones occluded from the depth sensor by being on the backside of the object. Even if the target voxel gets missed by the robot because of some mechanical inaccuracies, the contact will still occur in the backside of the object where the model will most likely be uncertain about the shape.

The issue with the current definition of variance lies in the fact, that it is independent of the weights, meaning the variance will stay unchanged after adjusting the weights. Since changing the weights is the only way this algorithm will influence the hypotheses, the unweighted variance as a metric will always target the same voxel. To fix this issue, the weights will be considered frequency weights to include them in the calculation of the variance. Given the weights $\omega = (\omega_1, ..., \omega_N)$ of the hypotheses $\mathbf{S} = (S_1, ..., S_N)$, the weighted mean μ_ω gets calculated by Equation 7. Using the weighted mean μ_ω and the weighted hypotheses, we can calculate the weighted variance σ^2_ω by

$$\sigma^2_\omega(v) = \frac{1}{\sum_{i=1}^{N} \omega_i} \sum_{i=1}^{N} \omega_i \cdot |S_i(v) - \mu_\omega(v)|^2. \tag{9}$$

Including the weights in the calculation of the variance has the important advantage of making the variance dependent on the weights, which are the only thing about the hypotheses the algorithm can influence. Since the weights do not update randomly but with the particular goal of matching the contacted voxel, it is to be expected that weights of disagreeing hypotheses get lowered in favour of agreeing hypotheses. For example, if the target voxel was contacted and confirmed occupied, the optimization would disregard most of the hypotheses that list the voxel as empty by reducing their weights. Besides fixing the value of that voxel, this also lowers the variance of it, because only similar hypotheses retain a significant weight. If the variances get calculated again for the next touch, the old target voxel will unlikely be chosen again. Instead the voxel with the highest uncertainty after the first touch will be targeted by the next touch.

3.2.2 Reducing maximum local entropy

The second metric is similar to the variance metric. Instead of calculating the variance of each voxel, the target voxel gets chosen by some form of entropy, which shares some properties with the variance. Originally entropy is defined as a measure of

expected information gain of a source. Given an event or source with n possible states $X = (x_1, ..., x_n)$, the entropy H is defined as the sum of the products of the probabilities p_i and information gain $I_i = -log_2 p_i$ of state x_i

$$H = \sum_{i=1}^{n} p_i \cdot -log_2 p_i \tag{10}$$

Finding the voxel with the highest entropy can be viewed as finding the voxel with the highest expected information gain. In the strict sense a voxel only has two states: occupied or empty. Calculating the entropy of one voxel would only involve the probability that the voxel is empty and the probability that it is full. After combining all hypotheses into one mean shape, all voxels hold a value from 0 to 1, which can be considered the probability that the voxel is occupied. In fact, this assumption is used when thresholding the voxels of the mean shape at 0.5 to obtain voxels of value 0 and 1 only in the final shape. Since the probability of an empty voxel is simply 1 minus the probability of an occupied voxel, the mean value of all hypotheses would be enough to calculate the entropies of the voxels.

While this approach would follow the strict definition of entropy, it contains two problems. First, while the hypotheses combine nicely into voxels of continuous values from 0 to 1 which could be considered probabilities, the hypotheses were not generated on a probabilistic basis. They were generated to minimize a loss function, therefore using the resulting values as probabilities would already violate an assumption of entropy. The second problem lies in the simplicity by which the entropy would get maximized. Entropy for an event with only two outcomes is maximized when the two outcomes are of equal likelihood, meaning when the combined voxel has a value of 0.5. Using this metric the algorithm would simply target these 0.5 value voxels, only considering the mean of the hypotheses and not the distribution of weights and values.

To illustrate the issue, imagine 2 voxels, both of which have values proposed by two equally weighted hypotheses. For the first voxel, both hypotheses propose a value of 0.5, while for the second voxel one hypothesis proposes 1 and the other 0. Using the approach mentioned above, both voxels would have the same entropy, because it only depends on the mean value which is 0.5 for both voxels. Even though they result in the same entropy, the second voxel is clearly the more valuable target. Touching the voxels and obtaining their true value would be meaningless for the first voxel, because even though we would then be aware of the true value, there is nothing we can change to achieve that value in the combined shape. The only thing we can change are the weights of the hypotheses and since they both propose value 0.5, no matter the weights the resulting value will always be 0.5. The contrary is true for the second voxel. Finding out the true value allows us to disregard the false hypothesis by reducing its weight and assigning it to the correct hypothesis. Therefore, using just the mean value to calculate the entropy of each voxel is insufficient to identify interesting target voxels.

Instead, a metric is proposed that closely resembles the entropy presented above, but uses different states to compute it on. The possible states of a voxel are expanded from the discrete values zero and one to the continuous range between them. To obtain

the probability of each state, the hypotheses will be used as weighted observations. For each voxel, these observations can be represented by a histogram that orders the hypotheses by their value for the voxel in question. To calculate entropy from a histogram with discrete values over a continuous range, the observations need to be categorized in so called bins. These bins sort the hypotheses into groups of similar values. For example, if three bins were chosen, all hypotheses that have a value from 0 to 1/3 will be summed in the first bin, all hypotheses with a value between 1/3 and 2/3 will be summed in the second and all the remaining hypotheses will be summed in the third bin. Given a voxel v and m equidistant bins $\mathbf{B} = (B_i, ..., B_m)$, the values for each bin are the sum of weights of the hypotheses that fall into that bins range R:

$$B_i(v) = \sum_{j \in R} \omega_j \ \textit{with}\ R = \{k | \frac{i-1}{m} \leq S_k(v) < \frac{i}{m}, S_k \in \mathbf{S}\} \tag{11}$$

These bins can then be used to calculate the entropy H of a voxel through

$$H(v) = \sum_{B_i \in \mathbf{B}} B_i(v) \cdot -\log_2 B_i(v) \tag{12}$$

This metric is useful as a measure of disagreement between the weighted hypotheses. The two example voxels mentioned above now resolve to an entropy of zero for the first voxel and a high entropy for the second voxel. The exact value depends on the choice of number of bins, which will be further discussed in Section 4.3. Both this and the variance metric measure the disagreement between the hypotheses while also taking the weights into consideration. The difference between the metrics lies in how they value the absolute difference between disagreeing hypotheses. The variance depends on the exact difference, while the entropy only cares if the hypotheses fall into the same bin or different ones. If they belong to different bins, it does not matter how far apart those bins are.

3.2.3 Minimizing expected global entropy

The final metric tries to minimize the entropy of the complete voxel grid and will be referred to as the "global entropy approach". This would be the theoretical approach when dealing with entropy, but also the most computationally expensive one. It tries to predict which target voxel will lead to the biggest global decrease in entropy and chooses accordingly. The global Entropy H_{all} is simply the sum of all individual voxel entropies

$$H_{all} = \sum_{v \in \mathbf{V}} H(v) \tag{13}$$

The way to predict the global entropy change for a given target voxel is to simulate a contact at that voxel and update the weights given that contact location. The result will be a new distribution of weights with a new different global entropy. By repeating this for every voxel on the surface voxel, the expected global entropy can be obtained for every possible target voxel. The target voxel will be chosen according to $\min_{v_t} E[H_{all}]$, resulting in the lowest expected global entropy.

This approach is very computational expensive, because it needs to simulate a touch and calculate optimal weights for every single voxel on the predicted shape's surface. It is possible to get more accurate predictions about the change in global entropy, but only at the expense of even more computational complexity. The approach presented above assumes a contact on the predicted surface and simulates everything based on that assumption. However, there is no guarantee that the chosen target voxel is actually occupied by the real shape. Therefore, the contact may occur before or after the target voxel, or even never at all. This leads to a difference between the real change in entropy and the predicted one. One way to reduce that difference is to not simply simulate the target voxel as a contact, but to simulate the path taken to that target voxel and consider the possibilities of the voxels in the path being occupied by the real shape. Since each voxel is represented by a value between 0 and 1, we can take that value as the probability of that given voxel being occupied. Now the change in entropy can be calculated for each voxel along the approach vector being the contact point and multiplying that change with the probability of that voxel being the real contact. The sum of these probability weighted changes in entropy results in the expected change in entropy. While this may lead to a better estimation of the real change in entropy, it is even more computationally expensive, because for every voxel on the predicted surface, multiple contacts have to be simulated and evaluated. Specifically, one for every voxel visited in the approach path, which roughly amounts to 40 voxels per path, meaning 40 times the computational cost. This improved, even more expensive approach will not be implemented because it would take too much time to evaluate it in a simulation or use it in a real-world application.

3.3 Approaching target voxel

Approaching and touching the target voxel is how information about the real shape is obtained. That information includes where the contact occurs, as well as which voxels the robot end-effector passed through before contacting the object. The contact location can be extracted using forward kinematics and converted into voxel space, resulting in a voxel which is confirmed to be occupied, called the contact voxel v_c. All the voxels passed before that are confirmed to be empty and will be saved in the set of path voxels $\mathbf{V}_{path}$.

$$occ(v) = \begin{cases} 0, & v \in \mathbf{V}_{path} \\ 1, & v = v_c \end{cases} \tag{14}$$

The exploration path, or the approach, will be a straight line that passes through the target voxel and aims to be oriented perpendicular to the surface at the target voxel.

3.3.1 Perpendicular approach path

All three proposed exploration strategies will only focus on selecting a target voxel and approach that target voxel in a perpendicular line. The alternative to a perpendicular line would be a more complex path that tries to optimize any given metric. However,

because there are virtually infinitely many lines and even more paths leading to a single voxel, this would be computationally unfeasible and impractical. Out of all the possible lines that pass through the target voxel, a perpendicular one offers the highest robustness to inaccuracies in the predicted shape. The predicted shape is only our best estimation of the real object. There is no guarantee that the voxel we try to contact will actually be occupied by the real object. Moving perpendicular to the predicted shape, however, will still yield a contact most of the time.

It is neither possible nor necessary to find a perfectly perpendicular line because the shape is approximated by voxels and an almost perpendicular line still holds the advantages mentioned above. Only the voxels in close proximity to the target voxel will be used to calculate the approach vector. The neighbourhood radius defines how far away a voxel can be from the target voxel to still be taken into account.

A voxel v has the position $L(v) = (x, y, z)$. Let us call the target voxel v_t and the set of all occupied voxels that are inside the neighbourhood $\mathbf{N}$ of v_t. The approach vector $\vec{a}$ is then calculated by

$$\vec{a} = \sum_{v \in \mathbf{N}} L(v) - L(v_t)$$

There is no clear choice for the neighbourhood radius, as choosing a small value will lead to a low resolution in the approach vector and possible over weighting of corner voxels that only represent a slightly tilted surface. Choosing too high a value will distort the line for curved surfaces because far away voxels are less likely to have the same surface angle as the target voxel. A neighbourhood radius of 2 was chosen, meaning all voxels that differ by no more than 2 in any given dimension from the target voxel were taken into the calculation.

Figure 4 visualizes how the perpendicular line gets calculated. The red pixel is the target pixel, the green pixels form the predicated shape and the highlighted ones are in the target pixel's neighbourhood. The blue arrows show the individual vectors which together form the approach vector shown in black. The example is kept in 2D with a neighbourhood radius of 1 while the real case has 3 dimensions and uses a neighbourhood radius of 2. The concept stays the same, but is easier to understand in the simpler case.

3.3.2 Real-world application vs simulation

This step in the algorithm is the only one that would require interaction with a real-world object, meaning this is where measurement errors would be introduced in real applications. While the simulations will provide exact and reliable data, the real-world application will be affected by mainly two error sources. Firstly, the inaccuracies in the location obtained through forward kinematics. Because of slack in the joints or inaccuracies in the robot model, the location obtained through forward kinematics will be of limited precision. A second, bigger error source is movement in the object upon contact with the end-effector. While the end-effector will stop as soon as possible upon registering a contact, it will inevitably move a little bit further than the perfect surface. Besides translating and rotating the object a little bit,

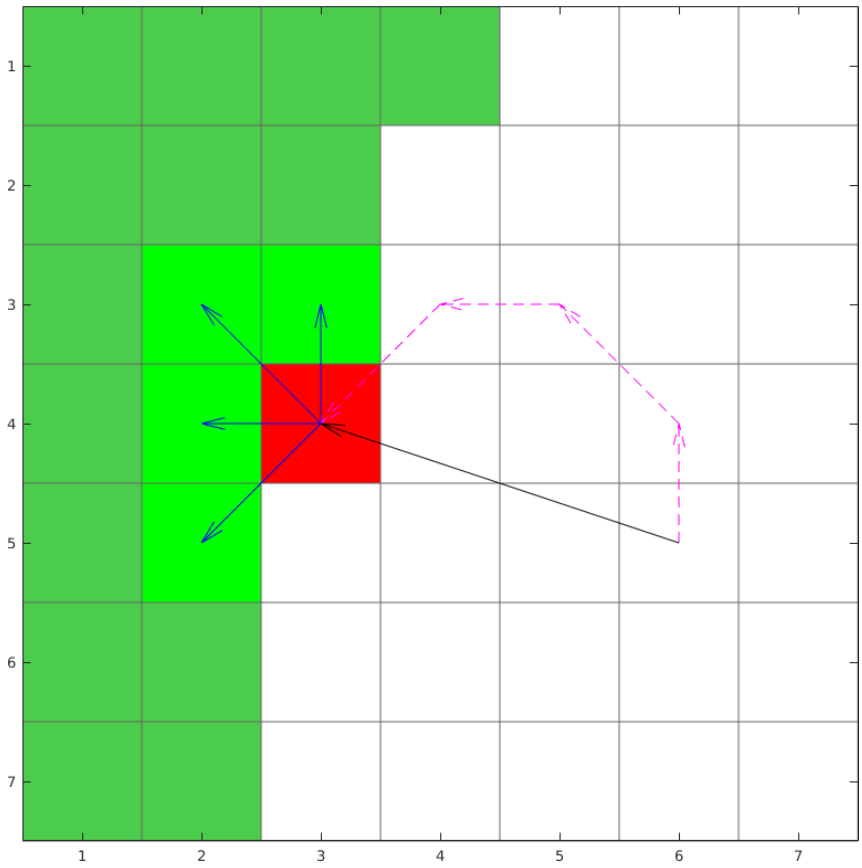


Figure 4: Perpendicular line visualized for 2 dimensions and a neighbourhood radius of 1.

depending on weight, height and friction of the object, contacting it in an unfortunate location may even result in it tipping over. While precautions will be taken to reduce object movements, it is impossible to eliminate them completely. The introduced inaccuracies might compound with every touch, if the object keeps moving. All of these errors are of no concern in the simulations and for this reason outside the scope of this thesis.

Additionally, the simulations have a method of identifying the contact voxel and the empty voxels passed beforehand that would be impossible in real world applications. The exact method will be explained in Chapter 4, but on a real robot the method would have to be changed to the following: The real-world robot will use an end-effector covered with force sensors to approach the object according to the target voxel and approach vector. Once the force sensors register a contact the robot will stop moving. Using forward kinematics, the contact location in real-world coordinates can then be calculated from the robot joint states. Given robot and camera transformations, the real-world coordinates can be converted into voxel coordinates, yielding the contact voxel.

3.4 Updating belief after contact

The final step in the algorithm is using the confirmed occupied contact voxel v_c and the confirmed empty voxels in the path $\mathbf{V}_{path}$ to calculate better fitting weights for the hypotheses. Since this algorithm allows for iteration, meaning it can be applied multiple time to use more than one touch as observations, the first step is to combine the information gained in this iteration with the previous iterations. If this is iteration number i, then the set of all voxels observed in this and previous iterations will be denoted as the set of partial truth voxels $\mathbf{T}_i$

$$\mathbf{T}_i = \mathbf{T}_{i-1} \cup \mathbf{V}_{path} \cup v_c. \tag{15}$$

Since the true values of voxels in the partial truth are known through the exploration, the weights can be optimized to match the resulting values $res(v)$ of these voxels to the true values. In other words, the weights will be chosen so that the combined and thresholded shape is as similar to the true values of the known voxels as possible. The metric to measure the similarity between the true values and the combined shape will be the Jaccard Index. The Jaccard Index is a similarity measure between two sets and is calculated by the intersection over the union of the two sets,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{16}$$

where A and B are the two sets to calculate the Jaccard Index on. The Jaccard Index can be interpreted as the percentage of matching elements in the two sets.

Using this Jaccard Index between the resulting shape values and the true values of the partial truth voxels the choice of weights will aim to maximize the similarity.

$$\max_{\omega} J(occ(\mathbf{T}_i), res(\mathbf{T}_i)) \tag{17}$$

That way, the resulting shape will match the observations optimally according to the Jaccard Index. The only variables in this Jaccard Index are the weights ω_i of the hypotheses which are constrained from 0 to 1, allowing this nonlinear optimization problem to be solved by sequential quadratic programming (SQP).

3.4.1 Normalized weights vs unnormalized weights

One design choice is whether the weights should be normalized or not. While all weights can have a value from 0 to 1 ($\omega_i \in (0, 1)$), normalizing weights mean adding the constraint that the sum of all weights should always be 1

$$\sum_i \omega_i = 1. \tag{18}$$

While this constraint is intuitive when interpreting the weights as percentages, it also removes one degree of freedom in choosing the optimal weights. It is hard to predict how the algorithm performs differently when changing from normalized weights to unnormalized ones, where each weight can have a value from 0 to 1, independent of

the other weights. Unnormalized weights have the clear advantage of offering a bigger possibility space, since all combinations of normalized weights are included in all combinations of unnormalized ones. One simple example would be three hypotheses that all share the same main body, but each hypotheses has a distinct extrusion unique to them. If all three extrusions plus the main body were part of the real shape, normalized weights would offer no possible combination to have all three extrusion included in the combined shape. Unnormalized weights however could simply assign the value one to each hypothesis and have the combined shape be the union of all three. Normalized weights on the other hand, enable the changing of one weight to have a direct effect on all others and can for example force one region to be empty which is not possible without normalizing the weights. Furthermore, normalized weights follow most conventions which allows them to act as probabilities. This becomes important when calculating the entropy of voxels in the first step of the algorithm.

4 Simulations

This chapter will describe the simulations carried out to evaluate the effectiveness of the proposed approaches. It is divided into four sections: The first section briefly explains on which dataset the algorithm was evaluated and how the dataset can be categorized. The second section will introduce a step-size δ, which can determine confidence of single iterations. The optimal value for this parameter is hard to predict, due to the unpredictable weight updating process. Therefore, the simulation will be executed on multiple step-size values and the choice for the optimal value will be based on the emerging results. The final section will describe the implementation by following the information pipeline and justifying design choices made.

4.1 Dataset

The algorithm was tested on a total of 126 models from the YCB [37] and Grasp Database [38] mesh model datasets. These 126 models were obtained by placing 63 different objects in varying number of poses and matching the ground truth of each object to the current pose. The 126 models can be divided into the following three categories: trained view, holdout view and holdout model. Models in the trained view category were used to train the NN that generates the hypotheses. That means the NN had seen the object in that pose before and was also presented with the corresponding ground truth before. Models in this category generally yield well-fitting hypotheses. The second category, holdout views, contains models of objects that were also used in the training of the NN, however they are arranged in novel poses. The novel poses force the NN to interpolate from its trained data, but since all the features were included in the training data, the resulting hypotheses are similarly accurate as in the trained view category. The final category, holdout models, can be considered the most relevant one, as it consists of objects the NN has never seen before. Since the NN has to interpolate from features of different objects, the resulting hypotheses will be significantly less accurate as in the other two categories. Nevertheless, the majority of objects in real-world applications will fall into this category. Therefore, this category will be the focus in the summary and discussion of the results.

4.2 Step-size

In the weight-adjusting step of the algorithm a complete new set of weights gets proposed for the hypotheses. While these new weights achieve maximal similarity with the observed voxels, it is unclear whether all hypotheses should completely disregard their old weights to adopt the new ones. The new weights are based on a sparse observation and just disregarding the previous weights might result in overconfidence. One drastic example would be a contact voxel which is only occupied by a single hypotheses. In that case that hypotheses would get a weight of one with all others getting a weight of zero. At that point the exploration would end in two of the three approaches because the entropy has reached zero and the model is confident

about the shape. If a perfect hypothesis existed, this confidence would be a good thing since the perfect hypothesis would never be disregarded. However, since we have to assume that the perfect shape cannot be modelled from the hypotheses, much less from a single hypothesis, a better approach may be to limit that confidence and allow for further exploration to correct the unfortunate previous contact.

To achieve this, we introduce the step-size parameter δ, which determines how much influence the old weights should have when adjusting to new weights. Let ω_o be the old weights, ω_p the optimal weights according to the new observation and ω_n the new weights after taking both the old as well as the optimal weights into consideration. The new weights ω_n get calculated by

$$\omega_n = \omega_o + \delta(\omega_p - \omega_o). \tag{19}$$

That way the step-size parameter can be used to limit the change in weights and therefore confidence after a single touch. On the other hand, it will slow down the progress towards the optimal weights, which is the reason why the algorithm will be tested on different values.

4.3 Implementation

Inputs

To perform the algorithm in simulations, two inputs are required: The hypotheses and the initial weights of the hypotheses. Additionally the ground truth of the object will be supplied to evaluate the resulting shape after the simulation. For each object, 100 hypotheses were generated by the NN developed in [5] as mentioned in Section 2. The number of hypotheses was chosen to balance the possibility space of the combined shape with the computational cost connected to optimizing more weights. At around 100 hypotheses, their quality starts degrading by not offering any new features. At this point more hypotheses would only increase the dimensions of the parameter space that needs to be optimized later on, without offering significant benefits.

For the initial weights, all hypotheses were considered equal, resulting in initial weights of $1/100 = 0.01$. Only after the algorithm has been applied at least once can the individual hypotheses be evaluated and assigned individual weights. The ground truth is required to identify where a contact would occur when approaching from a given direction. Obviously, no ground truth will be available in the real-world application, instead an actual object would be present that can be touched by a robot. The ground truth will also be used to evaluate the resulting shape in terms of Jaccard Index.

Metric parameters

Depending on the approach for selecting the target voxel, the first step is either to calculate the variance or the entropy of every single voxel. In case of the variance, no design choices remain, as it is defined as in Chapter 3 and can easily be calculated

by an existing function in matlab. The entropy however has the remaining choice of how many bins should be used to create the histogram. These bins are used to categorize the hypotheses depending on their opinion on a voxel. The intuitive choice of three bins was selected, which means that the hypotheses can be categorized into "empty", "uncertain" and "full". The resulting ranges are 0 to 1/3 for the "empty" bin, 1/3 to 2/3 for the "uncertain" bin and 2/3 to 1 for the "full" bin. Having only three bins makes it easier for the algorithm to reduce the entropy after a touch than having a lot of bins. This is useful, because it means a touched voxel will more likely have a low entropy and will not be selected as a target voxel again.

Instead of minimizing local entropy or variance, one approach tries to minimize the expected global entropy through each touch (see Section 3.2.3). For that purpose, the global entropy is defined as the sum of all local entropies described above and the expected global entropy change gets simulated by substituting the current belief shape for the ground truth.

This approach is very computational expensive, because it needs to simulate a touch and calculate optimal weights for every single voxel on the predicted shape's surface. Most of the shapes in the dataset have over 1000 voxels on the surface, meaning over 1000 contacts have to be simulated and evaluated. It takes roughly one second to calculate the shape's entropy, simulate a single touch and update the weights, with the majority of that one second being spent on updating the weights in an optimal way. This leads to a total time of over 1000 seconds for this approach to figure out which voxel it wants to touch, making it unfeasible in a real-world application. Therefore this approach will only be tested on the dataset of holdout models with a step-size of 1 to compare the performance to the other approaches.

Surface voxels estimation

Because the choice for the target voxel gets limited to voxels on the projected (expected) surface, it is necessary to first extract the surface from the projected shape. As a remainder, the projected shape is the shape obtained by combining all hypotheses into the weighted mean shape and thresholding that combined shape at 0.5. A surface voxel then is defined as a voxel with value 1, which has at least one adjacent voxel with value 0. As adjacency definition, we consider 6-connectiveness. The only issue are voxels deeper inside the projected shape, that for some reason have a value of 0. This is unlikely, but because the hypotheses get generated by an unpredictable NN, not impossible. These empty voxels inside the projected shape lead to the six adjacent voxels being identified as surface voxels. However no heuristic was chosen to identify those "false" surface voxels, because they were considered vastly outnumbered by the real surface voxels and even if one of the false ones would get chosen as the target voxel, the only disadvantage would be a less informative touch, because it basically touched somewhere random.

Exploration path

After the surface and entropy have been determined the target voxel and approach vector can be calculated according to the approaches presented in Chapter 3. But

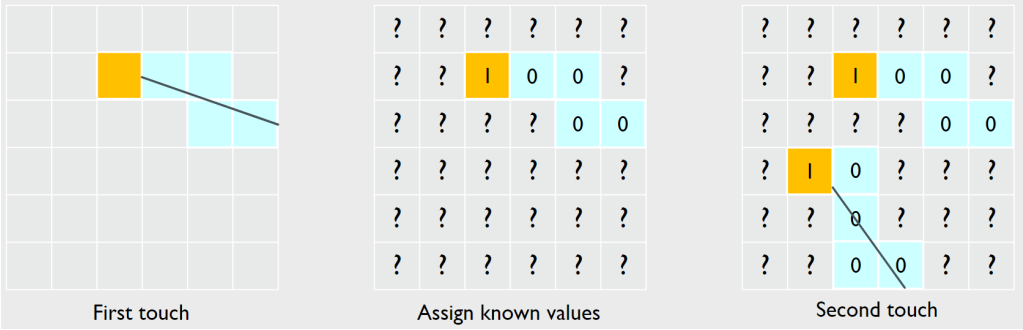


Figure 5: Iterative process of generating partial truth visualized in 2D.

the target voxel is part of the projected shape, not the real shape, which means there is no guarantee that the actual contact will occur exactly at the target voxel. To simulate where the actual contact would take place the ground truth will be used. The exploration path will be modelled as a line that begins at the starting voxel v_s which is the voxel on the edge of the 40x40x40 voxel space where the approach vector $\vec{a}$ needs to start in order to pass through the target voxel v_t. In other words, it is the intersection of the voxel space boundary box and the half-line created by $v_t - m\vec{a}$.

Given the starting voxel v_s and the approach vector $\vec{a}$, it is now possible to generate a line through the voxel space, recording every voxel the line enters, until a voxel is entered that is occupied by the ground truth. That voxel is where the real contact would occur and will be called contact voxel v_c.

Partial truth update

All of the voxels passed through in the exploration path will become part of the partial truth. Because these voxels were compared to the ground truth to see if a contact occurred or the end-effector would keep moving, the true value of these voxels are now known. The partial truth is a list of all these voxels, obtained from this touch and all the previous touches on the same object. That list contains the value and the location of all known voxels. It is possible, that one voxel is in the exploration path of more than one touch. But because no measurement errors or real-world inaccuracies apply in the simulation, the ground truth will always report the same value for that particular voxel and no conflicting information can occur. Figure 5 shows a two-dimensional example of what this step could look like. The partial truth is the list of voxels that have a confirmed value of either 0 (empty) or 1 (occupied).

Weight optimization

The list of partial truth values will be used as new information to calculate new weights for the hypotheses. As explained in Chapter 3, the weights will be chosen with the goal of maximizing the Jaccard Index of the partial truth voxels and the corresponding voxels of the combined shape. The optimization gets performed according to the sequential quadratic programming (SQP) method implemented by

the matlab function 'fmincon'. Fmincon offers a solver for minimizing constrained nonlinear multivariate functions, with the constrains determining whether the weights shall stay normalized or not. The SQP algorithm was chosen because it achieves higher accuracies than the default interior-point algorithm [39]. The drawback is that sQP requires more time and memory but running the algorithm on many objects was manageable, since for a single touch, the optimization has to be performed only once, which took approximately one second.

One problem occurs when using the Jaccard Index as the function that needs to be maximized. That problem is the fact that the Jaccard Index has zero gradient, meaning that small changes in the weights have no effect on the Jaccard Index. The reason for that lies in the thresholding of the combined shape. Only changes in weights that lead to a voxel moving over or under the threshold will also change the projected shape and therefore the Jaccard Index. If the change is too small, all voxels will still be in the same threshold category and the resulting shape will not change. Having zero gradient leads to the issue, that an optimization algorithm does not know how to change the weights in order to increase the Jaccard Index. That means the algorithm has to take random samples and simply check which sample yields the best result. While algorithms exist that sample more sophisticated than random, high dimensionality of the parameter space makes a reliable optimization infeasible without a gradient.

To solve that issue, a the parameter steepness δ is introduced to the thresholding. Instead of thresholding to 0 and 1, the voxels will be thresholded by a sigmoid function centred around 0.5 according to

$$output = \frac{1}{(1 + e^{-(input-0.5)\cdot\delta})}. \tag{20}$$

By running the optimization 40 times with δ increasing gradually from 10 to 7500 and using the resulting weights of the previous iteration as the initial weights for the next iteration, the optimization will occur on a function with non-zero gradient that approximates the Jaccard Index with increasing δ. Figure 6 illustrates how the thresholding behaves for different δ values. Every iteration of increased δ will move the weights more towards the optimal weights, while keeping a non-zero gradient. The reason for gradually increasing the steepness instead of just using the final value of 7500 is that even though 7500 has a non-zero gradient, it is so small that the optimization would progress almost as slowly as if the gradient was zero.

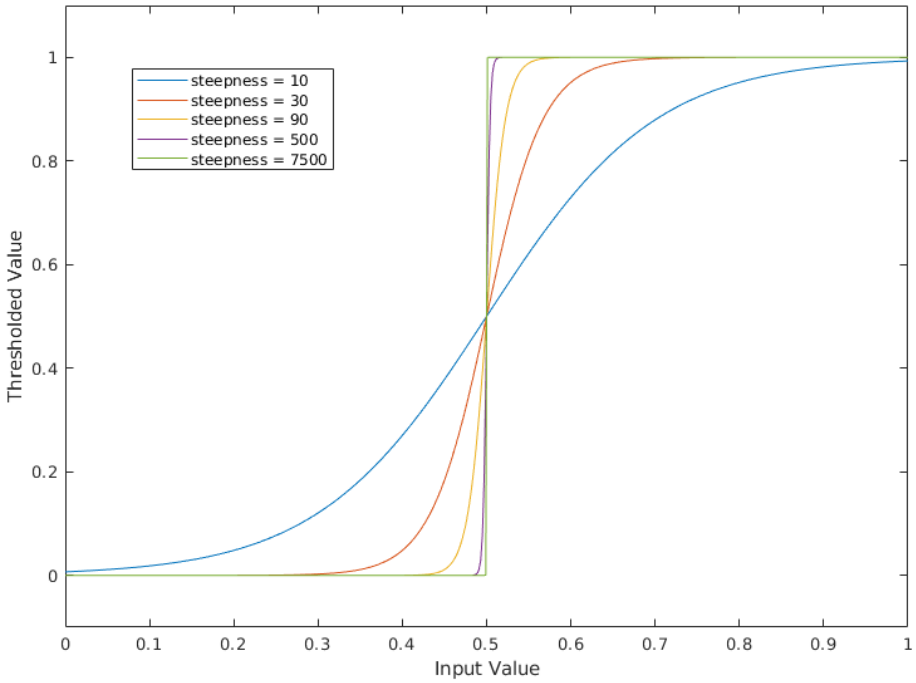


Figure 6: Gradual Thresholding through increasing steepness

5 Results

This chapter will present the results from the simulations. While the focus will be on the resulting Jaccard Indices, the effect of the algorithm will be presented in detail on some example objects. These experiments aim to answer the following questions:

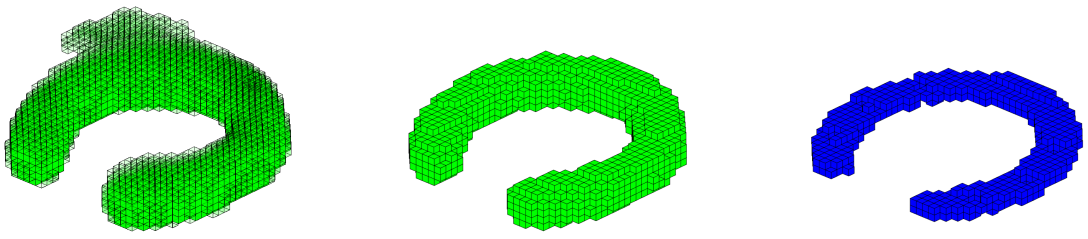
- How does normalizing the weights change the performance compared to unnormalized weights and can one model be considered superior to the other?
- Do the three different datasets (trained views, holdout views, holdout models) result in different performances?
- How does the global entropy approach compare to the local entropy and local variance metric?
- Does the step-size parameter affect the performance. If yes, can an optimal value be identified?
- Which values of the Jaccard Index can the proposed method achieve compared to the limits set by the underlying hypotheses?

The complete results can be found in Appendix A

5.1 Example Object

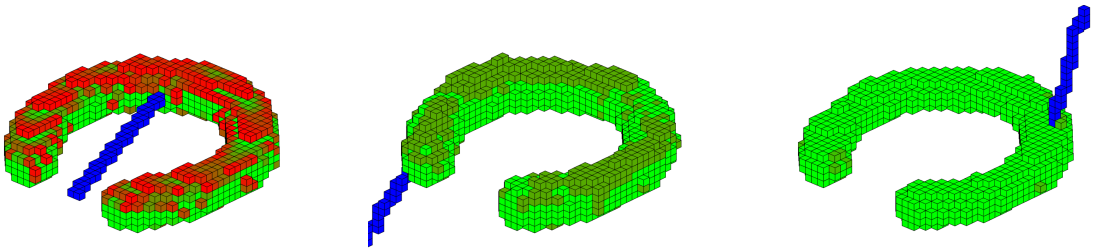
To visualize the procedure of the method, this section will present the progress of one example object during the algorithm. The chosen example object is a horseshoe from the holdout model dataset with unnormalized weights, step-size one and local entropy as the exploration method. It was chosen because it achieved a significant improvement of the Jaccard Index over multiple touches, without being perfect with every iteration, highlighting some strengths and weaknesses of the algorithm.

Figure 7 shows the real shape of the horseshoe (ground truth) and what the resulting shape looked like after combining the 100 hypotheses with equal weights, meaning before the algorithm was applied. The initial resulting shape has a Jaccard Index of 0.319 with the ground truth.

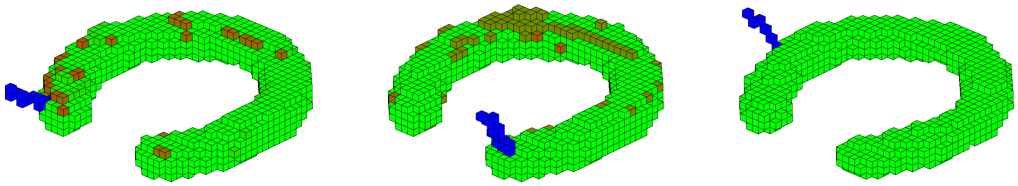


(a) Combined Shape (b) Resulting = Thresholded Shape (c) Ground Truth

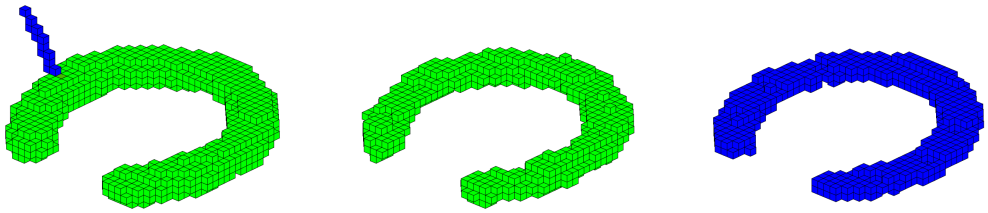
Figure 7: Initial horseshoe shape before the algorithm.



(a) 1st touch (b) 2nd touch (c) 3rd touch



(d) 4th touch (e) 5th touch (f) 6th touch



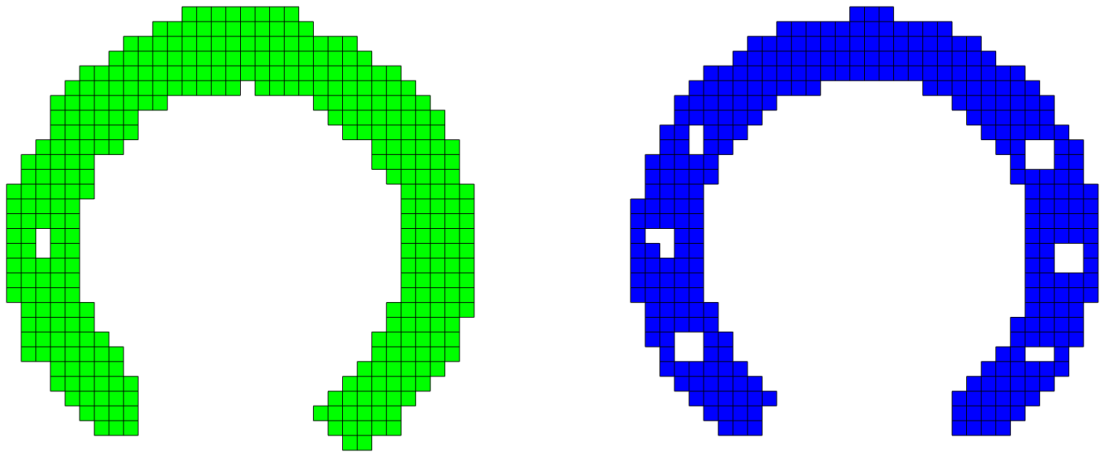
(g) 7th touch (h) Final Shape (i) Ground Truth

Figure 8: Iterations of the algorithm. Red Voxels represent high entropy, green voxels represent low entropy and blue voxels are used for the ground truth and the exploration paths.

Figure 8 visualizes the improvement achieved by the algorithm. Each picture shows the thresholded combined shape before each touch, with red voxels representing high local entropy and green voxels representing low entropy. The blue voxels show the exploration path taken, passing through the highest entropy voxel. The final two pictures show the final shape after the 7th touch and the ground truth for comparison.

The algorithm concluded after the 7th touch, because all surface voxels had local entropies below the threshold of 0.0001, indicating a high confidence of the model and leaving no good voxels to explore. The entropies can be seen to decline drastically over the first two touches, viszualized by the red and green voxels in Figure 8, but then increase slightly with the 3rd and 4th touch before being reduced to almost 0 with the last 3 touches. This shows that the entropy does not strictly decrease. The reason for this is that the optimization tunes the weights in favour of a high Jaccard Index and not a low entropy. Nevertheless, the overall trend is a decrease in entropy, especially with the first touch which showed the biggest decrease for almost every object.

The observed progression of the thresholded shape is a continuous thinning of the



(a) Final Shape (b) Ground Truth

Figure 9: Final shape and ground truth seen from the top-down perspective.

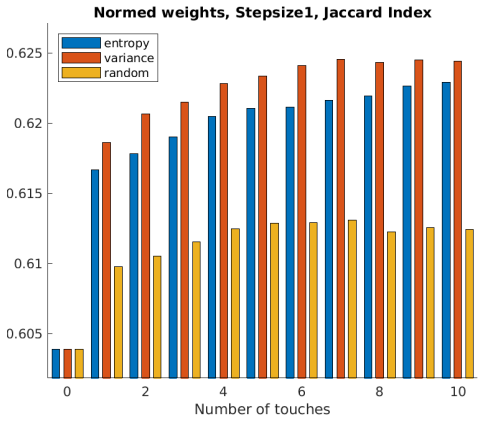
shape. When comparing the original shape with the ground truth (see Figure 7 the biggest difference is the thickness of the horseshoe. While the top-down perspectives may not match exactly, the biggest reason for the low Jaccard Index of 0.319 is the roughly 3 times too big thickness. The algorithm is able to reduce that defect through some touches, especially noticeable with touch 2 and 7. The final shape achieves a Jaccard Index of 0.617, which is one of the biggest increases of all the objects in the dataset.

Figure 9 shows the final shape and the ground truth from the top-down perspective. While not all voxels are identical, the shapes resemble each other very closely. One small success of the algorithm was finding one of the holes in the horseshoe with touch 7, allowing the weights to be adjusted in a way that includes this hole in the final shape.

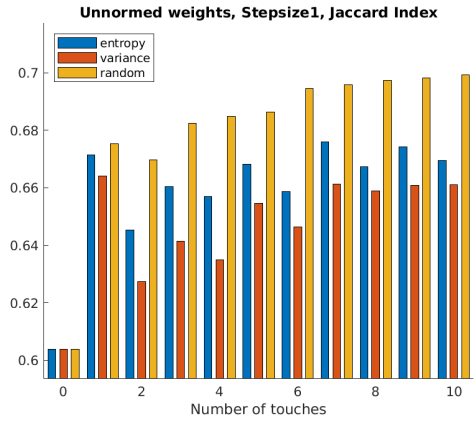
5.2 Normalized vs Unnormalized Weights

This section compares the results of the normalized weights approach with the unnormalized weights approach. For that purpose, the average Jaccard Index of all objects over 10 touches has been plotted in Figure 10. The three coloured bars represent the Jaccard Indices achieved by the local entropy approach, local variance approach and random exploration respectively. The exact values of some touches have been summarized in Table 1.

The two graphs in Figure 10 show drastically different results for unnormalized weights compared to the normalized ones. With normalized weights the algorithm seems to steadily improve with each consecutive touch. Furthermore, the local entropy and local variance approaches perform significantly better than the random exploration with the local variance approach slightly outperforming the local entropy approach. While the behaviour seems steady and well behaved, the overall improvement is only a gain in Jaccard Index by 0.019 for the local entropy and by 0.020 for



(a) Jaccard Index over 10 touches for normalized weights



(b) Jaccard Index over 10 touches for unnormalized weights

Figure 10: Comparison between normalized and unnormalized weights.

Touches	0	1	3	10
Entropy	0.604	0.617	0.619	0.623
Variance	0.604	0.619	0.622	0.624
Random	0.604	0.610	0.612	0.612

Normalized

	0	1	3	10
	0.604	0.671	0.660	0.670
	0.604	0.664	0.642	0.661
	0.604	0.675	0.682	0.700

Unnormalized

Table 1: Jaccard Index of different approaches

the local variance approach.

When allowing unnormalized weights the results differ in all properties mentioned above. The improvement is no longer steadily increasing with each touch but instead fluctuates around the initial improvement after the first touch, with no general trend visible. Many touches actually cause the Jaccard Index to decrease. Another difference is that the entropy and variance approaches no longer outperform the random exploration approach. Instead the random strategy is the only approach that seems to be able to improve after the first touch. Despite the irregularities the actual increase in Jaccard Index is much higher when compared to the unnormalized weights. After the first touch, the local entropy and local variance approach achieve a gain in Jaccard Index of 0.067 and 0.060 respectively. Compared to the gains of 0.013 and 0.015 of the normalized weights approach, the unnormalized weights achieve higher similarities.

What both approaches have in common, is that the first touch yields by far the biggest improvement, with all consecutive touches being significantly less effective or even destructive.

5.3 Dataset Comparison

This section compares the results between the different datasets, namely trained views, holdout views and holdout models. Figure 11 shows the results for the three

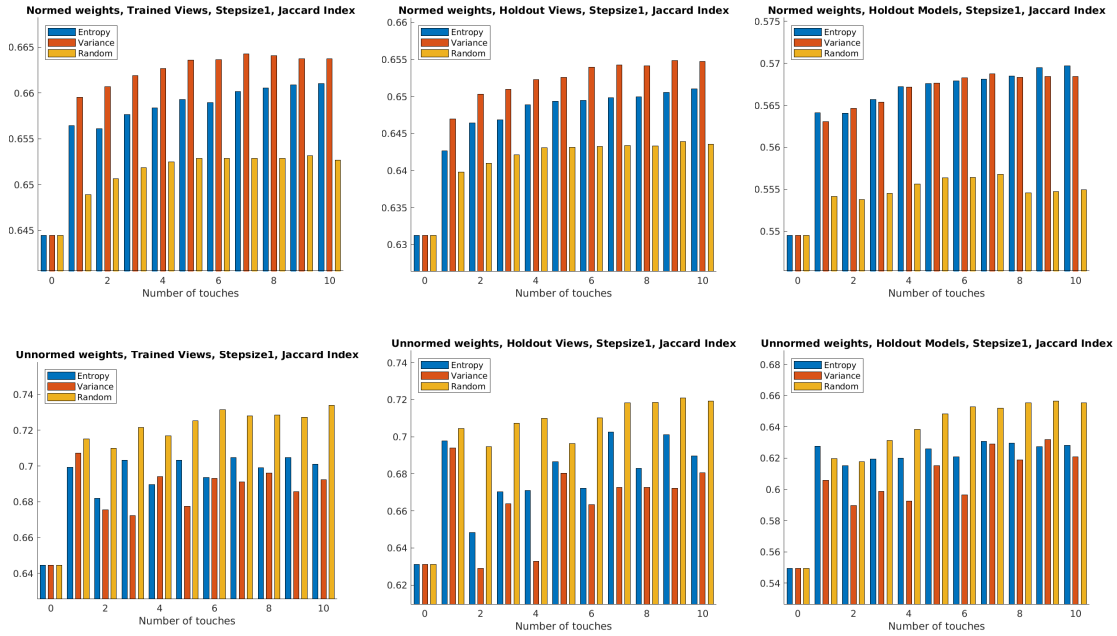


Figure 11: Top: Normalized weights, Bottom: Unnormalized weights.
Left: Trained Views, Middle: Holdout Views, Right: Holdout Models.

different datasets separated into normalized and unnormalized weights. Besides a lower baseline Jaccard Index for the holdout models no significant differences are visible for the different datasets. One minor difference is that the local entropy and local variance approach show similar improvements for the holdout models with normalized weights, compared to the local variance approach slightly outperforming the entropy approach in the other normalized weights datasets. Further results will focus only on the holdout models, since they represent the majority of objects in actual applications.

5.4 Global Entropy

This section presents the results from the global entropy approach. Since this approach was computationally expensive, it was only tested for the holdout models with normalized weights. Figure 12 visualizes the performance compared to the other approaches presented earlier. The first touch yields comparable results to the other two approaches but while they keep improving with consecutive touches the global entropy approach shows no visible improvements after the first touch. One possible reason for the stagnation of the global entropy approach may be the assumption, that a low entropy corresponds to a high Jaccard Index. Entropy is used as an indicator of highly informative contact locations, which does not automatically mean, that low entropy sections correspond to correctly combined hypotheses. Instead, forcing the model to reduce entropy as quickly as possible removes the model's only indicator for promising subsequent contact locations, without a more informative first contact.

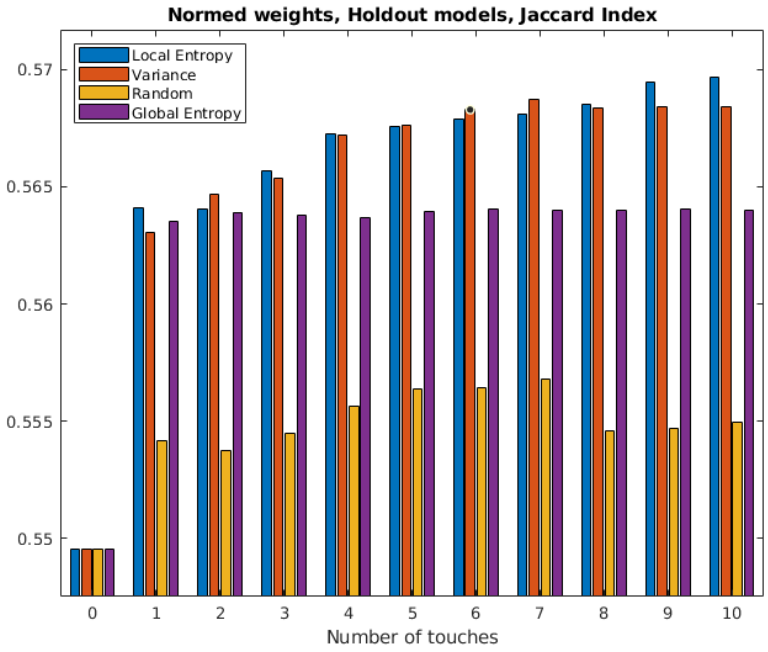
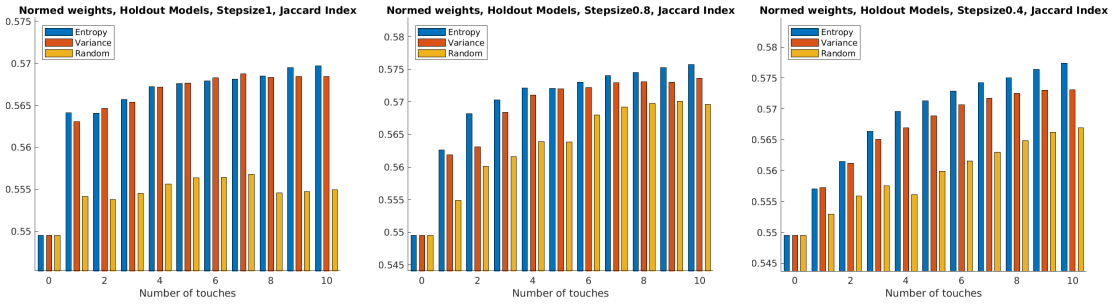


Figure 12: Global Entropy approach compared to the other approaches (step-size = 1)

5.5 Step-size

This section presents the results from different step-sizes. The idea with reduced step-sizes was to achieve more informative touches after the first one because of the reduction in possible confidence. Figure 13 shows the results for step-sizes of 1, 0.8 and 0.4 for the holdout models with normalized weights. The exact values for touches 0, 1 and 10 are listed in Table 2. Since the local entropy and local variance behave similarly, all summaries and conclusions are equally valid for both of them.

Results presented in earlier sections used a step-size of 1, which is why that step-size will be used as a baseline to compare the other two. Regardless of step-size and approach, the average Jaccard Index before exploration is 0.550. After the first touch a step-size of 1 achieves 0.564 (entropy approach), with step-sizes 0.8 and 0.4 achieving 0.563 and 0.557 respectively. The performance of the first touch drops as expected with lower step-sizes although the reduction from 1 to 0.8 is much lower than the reduction from 1 to 0.4. After ten touches, a step-size of 1 achieves 0.570, with step-sizes 0.8 and 0.4 achieving 0.576 and 0.577 respectively. As the step-size decreases a higher Jaccard Index becomes achievable through more than one touch. A step-size of 0.8 appears to be a suited compromise, because of the low drop in performance for the first touch compared to a step-size of 1 and almost the same improvement for 10 touches as the very low step-size of 0.4.



(a) Step-size = 1 (b) Step-size = 0.8 (c) Step-size = 0.4

Figure 13: Performance of different step-sizes.

step-size		1		0.8		0.4	
Touches	0	1	10	1	10	1	10
Entropy	0.550	0.564	0.570	0.563	0.576	0.557	0.577
Variance	0.550	0.563	0.568	0.562	0.574	0.557	0.573

Table 2: Jaccard Index of different step-sizes after 1 and 10 touches,

5.6 Potential

This final results section will compare the achieved Jaccard Indices to the limit that was achievable with the restriction of the given hypotheses. Because this algorithm only redistributes weights from given hypotheses, it is only possible to reconstruct shapes to the extend allowed by these hypotheses.

Table 3 shows the results achieved (local entropy and local variance approach averaged) in addition to a "∞-Touch" Value and an "Optimal" Value for each category. The "∞-Touch" Value is the average Jaccard Index that would have been achieved if the algorithm was given infinitely many touches. This value was calculated by extending the partial truth to all accessible voxels, meaning all surface and all empty voxels. The "Optimal" Value was the upper limit set by the hypotheses and corresponds to the Jaccard Index achieved through the best possible combinations of weights. It was calculated by replacing the partial truth with the entire ground truth.

These potential values reveal how effective the algorithm was by revealing the upper limit set by the hypotheses. For example in the normalized weights, holdout models category: While the improvement of the first touch seems only marginal (0.5623 from 0.5495), this improvement of 0.013 is almost 30% of the difference between the initial value (0.5495) and the perfect combination of weights for the 100 hypotheses (0.5927). The table reveals how much higher Jaccard Indices are possible for unnormalized weights, but also that these optimal values are harder to reach.

	Normalized weights			Unnormalized weights		
	Trained	HO-views	HO-models	Trained	HO-views	HO-models
Initial	93.9%	94.2%	92.7%	80.0%	78.0%	74.2%
1-Touch	95.6%	95.7%	94.9%	85.4%	85.1%	80.0%
10-Touch	97.2%	98.1%	97.0%	86.1%	87.3%	84.3%
∞-Touch	98.9%	99.8%	99.5%	94.5%	93.6%	91.0%
Optimal	0.6867	0.6702	0.5927	0.8081	0.8098	0.7403

Table 3: Average Jaccard Indices achieved (step-size = 0.8) as percentage of optimal weights (Optimal Jaccard Index shown in last row).

6 Conclusions

The goal of this thesis was to use sparse tactile glances to improve a shape generated only with partial information. For that purpose, different parameters were implemented and tested, consisting of the exploration strategy, normalizing weights and the stepsize.

For the exploration strategies, local entropy and local variance appear to yield almost identical results. Although the improvements made to the shape do not extend to new features, the Jaccard Index can definitely be improved with only a single touch. The improvements were limited to features already present in the hypotheses, leaving the question whether more diverse hypotheses would lead to even better results. The global entropy was only evaluated on a small dataset to see if it performed significantly better than the other approaches. While the first touch produced similar results to the other two approaches, the improvement stagnated for additional touches. Overconfidence in the model might be the reason for the stagnation, but since the approach is computational unfeasible, the entire algorithm would have to be changed to justify further research into that approach.

One very interesting choice was whether the weights should be normalized. The normalized weights behaved much more reliable and predictable than the unnormalized weights. However, the latter produced significantly higher Jaccard Indices which can partly be explained by the fact that unnormalized weights have one more degree of freedom, allowing for more possibilities. It is unclear, however, why the unnormalized weights resulted in such an irregular behaviour. If one model had to be chosen over the other, I would recommend the unnormalized weights. The reason being their high improvement rate through the first touch, which is by far the most interesting touch. A robot in a real-world application would rarely want to touch an object more than once before picking it up, due to the long time required for more touches. The first touch with unnormalized weights already results in the average Jaccard Index of the best possible (optimal in Table 3) normalized weight distribution, therefore outweighing the potential downside of unpredictable additional touches. Further research is recommended to better understand the unnormalized weights, in order to alter the model in a way that would make the unnormalized weights more reliable.

Another option to improve the performance of the algorithm could be to diversify the possible feature space by modifying the hypothesis generation. For this thesis the hypotheses were generated using a NN that was trained with a cross-entropy cost function, resulting in voxels values close to 0 and 1. It is possible that a more even distribution would allow for more gradual changes in the resulting shape, since changing a hypothesis' weight would then shift the object's boundary instead of enabling and disabling entire blocks. Further research is recommended to diversify the hypothesis generation.

The step-size parameter introduces a way to reduce the confidence of the model after only a few touches. This was useful to combat the problem of a single hypotheses attaining 100% of the weights after only a single touch. A step-size of 0.8 appears to be a good compromise between the fast improvement of a high step-size and the

higher long-term improvement of a low step-size.

Finally, all of the explorations were simulated in MATLAB, resulting in perfect contact voxel identification with no error sources that would be present in real world applications. These error sources include inaccurate forward kinematics and moving the object upon contact. Evaluating the algorithm's robustness to inaccuracies through real-world experiments remains a topic for future studies.

References

[1] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 2442–2447, iSSN: 2153-0866.

[2] S. Ottenhaus, L. Kaul, N. Vahrenkamp, and T. Asfour, "Active Tactile Exploration Based on Cost-Aware Information Gain Maximization," *International Journal of Humanoid Robotics*, vol. 15, no. 01, p. 1850015, Feb. 2018. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219843618500159

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1912–1920, iSSN: 1063-6919.

[4] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 2845–2850. [Online]. Available: http://ieeexplore.ieee.org/document/5980395/

[5] J. Lundell, F. Verdoja, and V. Kyrki, "Robust Grasp Planning Over Uncertain Shape Completions," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Mar. 2019, arXiv: 1903.00645. [Online]. Available: http://arxiv.org/abs/1903.00645

[6] M. Bjorkman, Y. Bekiroglu, V. Hogman, and D. Kragic, "Enhancing visual perception of shape through tactile glances," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, Nov. 2013, pp. 3180–3186. [Online]. Available: http://ieeexplore.ieee.org/document/6696808/

[7] J. Krivic and F. Solina, "Superquadric-Based Object Recognition | SpringerLink," in *Skarbek W. (eds) Computer Analysis of Images and Patterns. CAIP 2001. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2001. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-44692-3_17

[8] Scratchapixel, "Introduction to Polygon Meshes." [Online]. Available: https://www.scratchapixel.com//lessons/3d-basic-rendering/introduction-polygon-mesh

[9] A. E. Johnson and M. Hebert, "Control of Polygonal Mesh Resolution for 3-D Computer Vision," *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 261–285, Jul. 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077316998904749

[10] M. Ebden, "Gaussian Processes: A Quick Introduction," *arXiv:1505.02965 [math, stat]*, May 2015, arXiv: 1505.02965. [Online]. Available: http://arxiv.org/abs/1505.02965

[11] S. Gebhardt, E. Payzer, L. Salemann, A. Fettinger, E. Rotenberg, and C. Seher, "Polygons, Point-Clouds, and Voxels, a Comparison of High-Fidelity Terrain Representations," *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation—Processes, Standards, and Lessons Learned.*, p. 9, 2009.

[12] P. Bhowmick, "Partha Bhowmick: 3d Orthogonal Cover." [Online]. Available: http://cse.iitkgp.ac.in/~pb/research/3dpoly/3dpoly.html

[13] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the Art in Surface Reconstruction from Point Clouds," *Eurographics 2014-State of the Art Reports*, Apr. 2014.

[14] H. Hoppe *et al.*, "Surface reconstruction from unorganized point clouds," in *SIGGRAPH 92*, 1994, pp. 71–78.

[15] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, and S.-Y. Jin, *A Survey of Methods for Moving Least Squares Surfaces*. The Eurographics Association, 2008. [Online]. Available: https://diglib.eg.org:443/xmlui/handle/10.2312/VG.VG-PBG08.009-023

[16] D. Levin, "Mesh-Independent Surface Interpolation," in *Geometric Modeling for Scientific Visualization*, ser. Mathematics and Visualization, G. Brunnett, B. Hamann, H. Müller, and L. Linsen, Eds. Springer Berlin Heidelberg, 2004, pp. 37–49.

[17] P. Bremer and J. C. Hart, "A sampling theorem for MLS surfaces," in *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, Jun. 2005, pp. 47–54.

[18] Y. Lipman, D. Cohen-Or, and D. Levin, "Data-dependent mls for faithful surface approximation," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, ser. SGP '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 59–67. [Online]. Available: http://dl.acm.org/citation.cfm?id=1281991.1281999

[19] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Proceedings of the IEEE Visualization Conference*, 2001, pp. 21–28. [Online]. Available: https://nyuscholars.nyu.edu/en/publications/point-set-surfaces

[20] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales, "Mind the gap - robotic grasping under incomplete observation," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 686–693.

[21] A. H. Quispe, B. Milville, M. A. Gutiérrez, C. Erdogan, M. Stilman, H. Christensen, and H. B. Amor, "Exploiting symmetries and extrusions for grasping household objects," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3702–3708.

[22] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A Dataset for Improved RGBD-based Object Detection and Pose Estimation for Warehouse Pick-and-Place," *arXiv:1509.01277 [cs]*, Sep. 2015, arXiv: 1509.01277. [Online]. Available: http://arxiv.org/abs/1509.01277

[23] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3212–3217.

[24] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, Oct. 2003, pp. 474–481.

[25] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 998–1005.

[26] C. Papazov and D. Burschka, "An Efficient RANSAC for 3d Object Recognition in Noisy and Occluded Scenes," in *Computer Vision – ACCV 2010*, ser. Lecture Notes in Computer Science, R. Kimmel, R. Klette, and A. Sugimoto, Eds. Springer Berlin Heidelberg, 2011, pp. 135–148.

[27] C. Choi and H. I. Christensen, "3d pose estimation of daily objects using an RGB-D camera," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 3342–3349.

[28] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1886–1895.

[29] N. Sommer, M. Li, and A. Billard, "Bimanual compliant tactile exploration for grasping unknown objects," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6400–6407.

[30] D. Driess, P. Englert, and M. Toussaint, "Active learning with query paths for tactile object shape exploration," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 65–72.

[31] D. Watkins-Valls, J. Varley, and P. Allen, "Multi-modal geometric learning for grasping and manipulation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7339–7345.

[32] S. Wang, J. Wu, X. Sun, W. Yuan, W. T. Freeman, J. B. Tenenbaum, and E. H. Adelson, "3d Shape Perception from Monocular Vision, Touch, and Shape Priors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1606–1613.

[33] J. Ilonen, J. Bohg, and V. Kyrki, "Fusing visual and tactile sensing for 3-D object reconstruction while grasping," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3547–3554, iSSN: 1050-4729.

[34] A. Dai, C. R. Qi, and M. NieBner, "Shape Completion Using 3d-Encoder-Predictor CNNs and Shape Synthesis," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 6545–6554. [Online]. Available: http://ieeexplore.ieee.org/document/8100176/

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[36] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv preprint arXiv:1506.02158*, 2015.

[37] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and Model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics (ICAR)*, Jul. 2015, pp. 510–517, iSSN: null.

[38] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4304–4311, iSSN: 1050-4729.

[39] MathWorks, "Choosing the Algorithm - MATLAB & Simulink - MathWorks Deutschland." [Online]. Available: https://de.mathworks.com/help/optim/ug/choosing-the-algorithm.html#bsbwxm7

A Full Results

A.1 Stepsize = 1

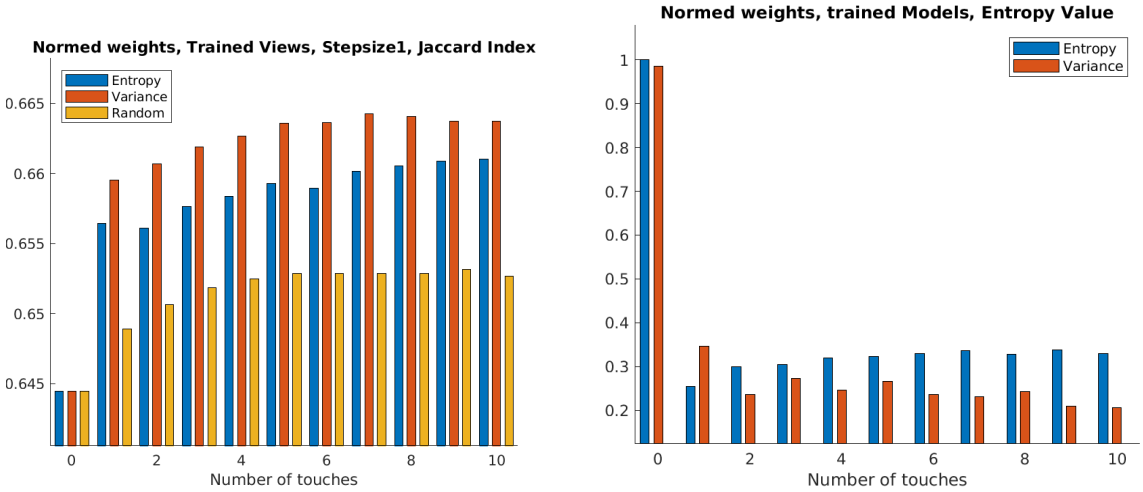


Figure A1: Normalized Weights, Trained Models, Stepsize = 1

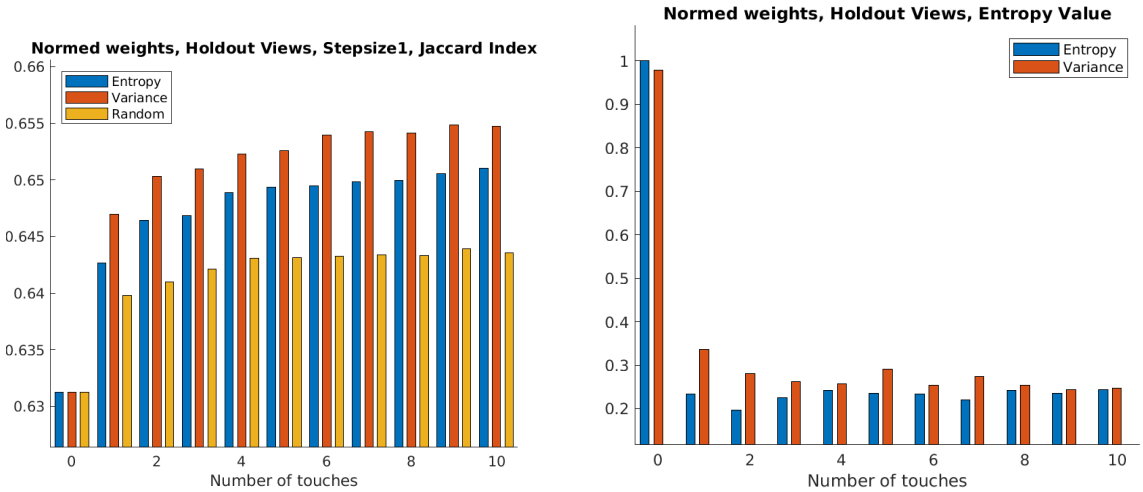


Figure A2: Normalized Weights, Holdout Views, Stepsize = 1

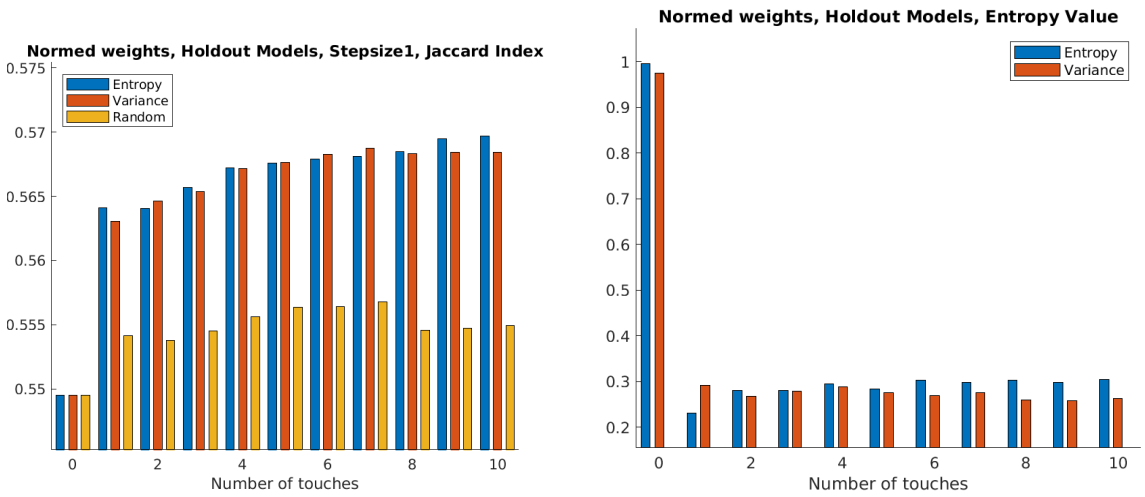


Figure A3: Normalized Weights, Holdout Models, Stepsize = 1

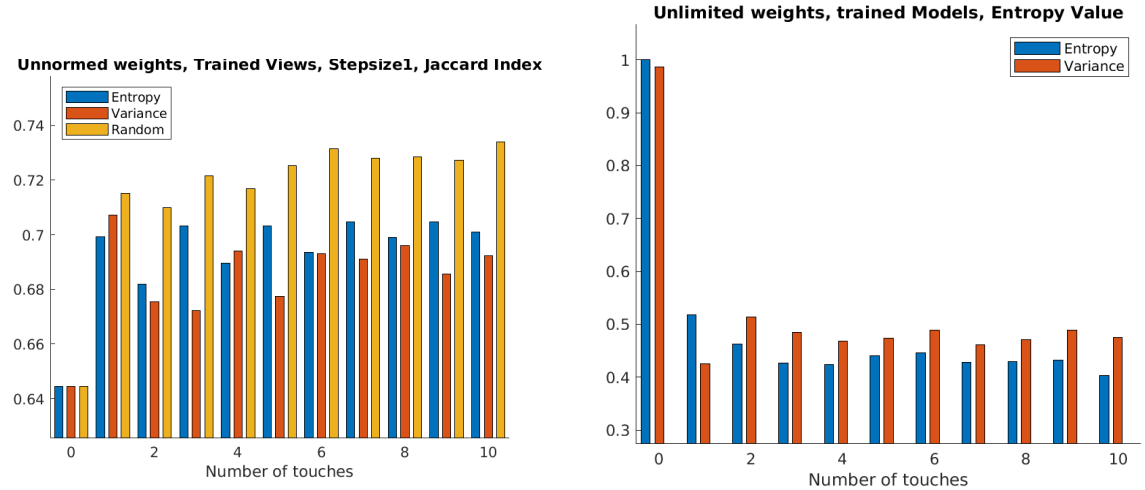


Figure A4: Unnormalized Weights, Trained Models, Stepsize = 1

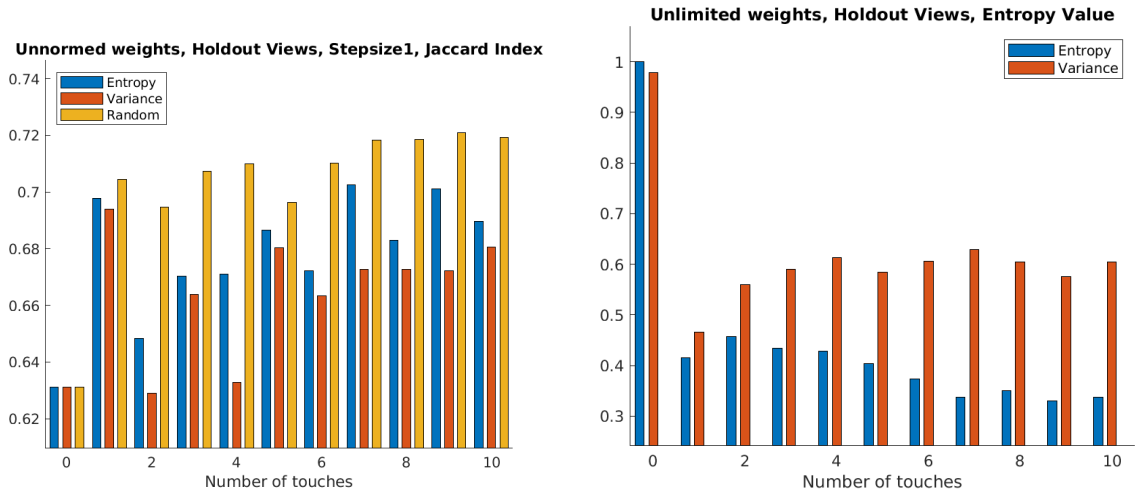


Figure A5: Unnormalized Weights, Holdout Views, Stepsize = 1

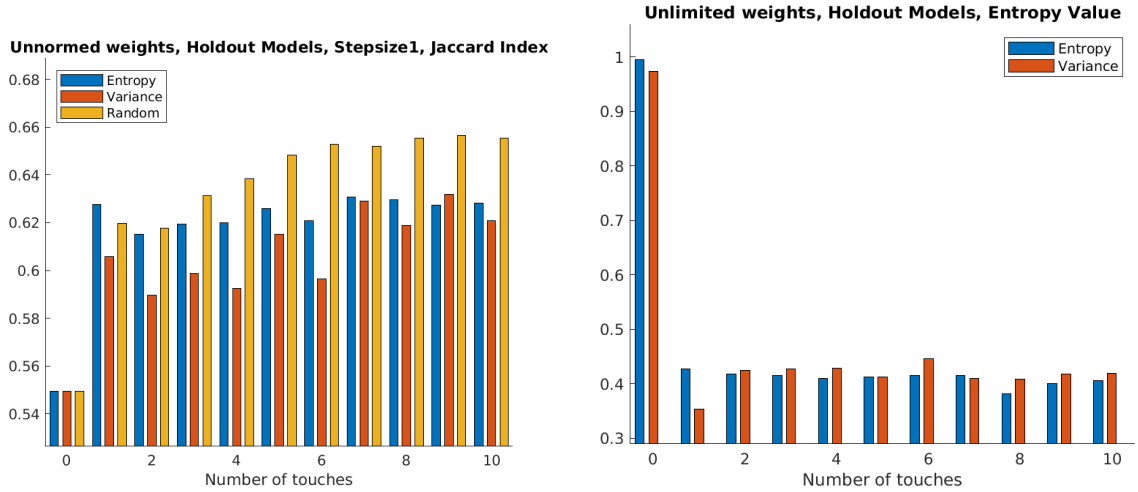


Figure A6: Unnormalized Weights, Holdout Models, Stepsize = 1

A.2 Stepsize = 0.9

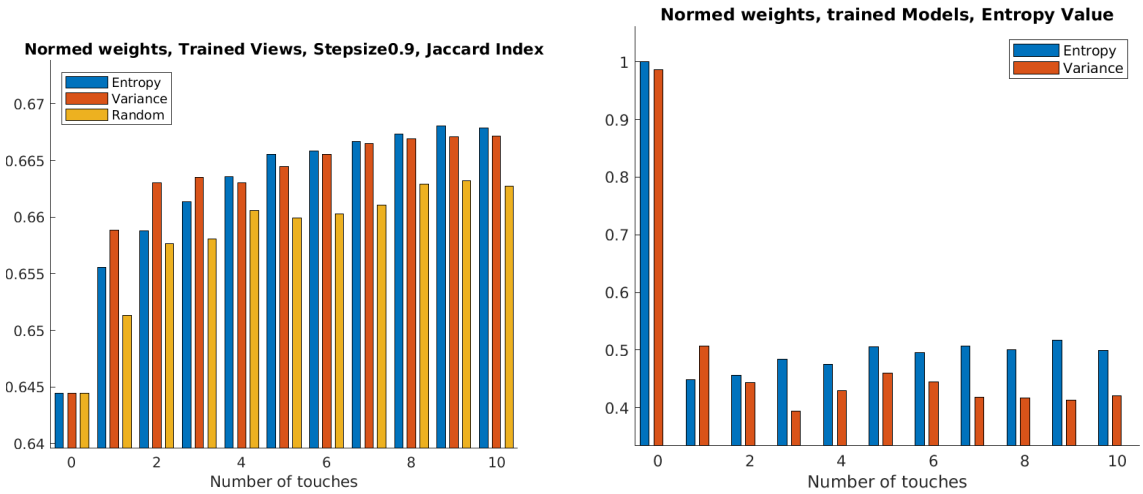


Figure A7: Normalized Weights, Trained Models, Stepsize = 0.9

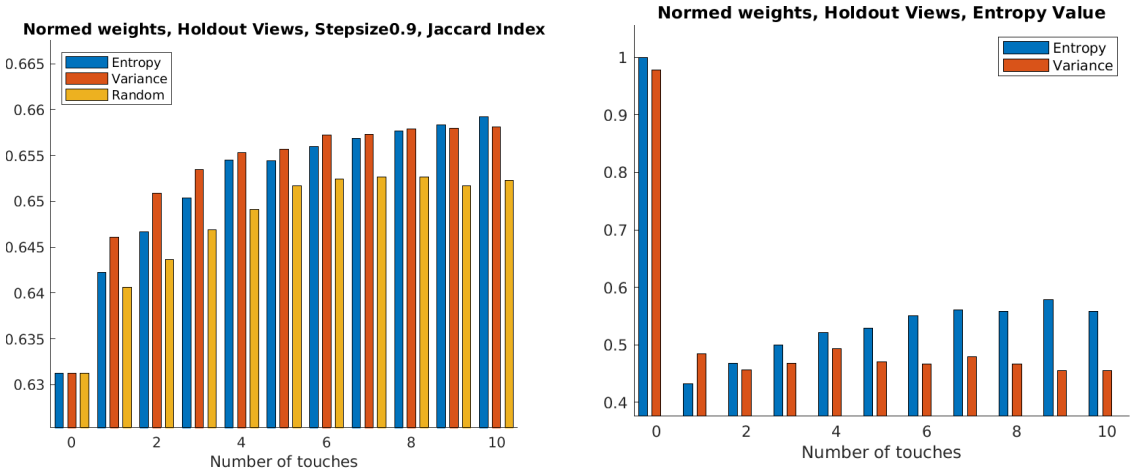


Figure A8: Normalized Weights, Holdout Views, Stepsize = 0.9

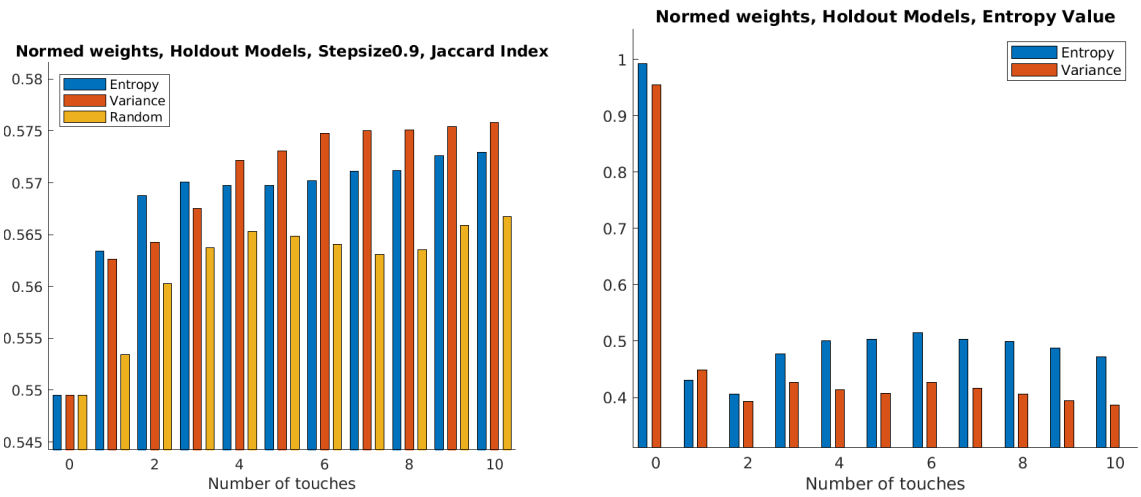


Figure A9: Normalized Weights, Holdout Models, Stepsize = 0.9

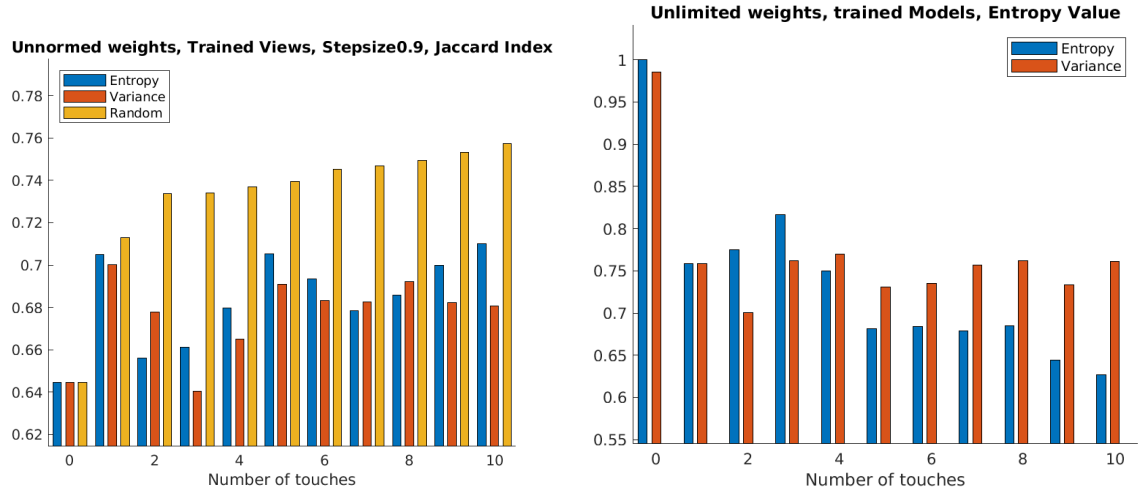


Figure A10: Unnormalized Weights, Trained Models, Stepsize = 0.9

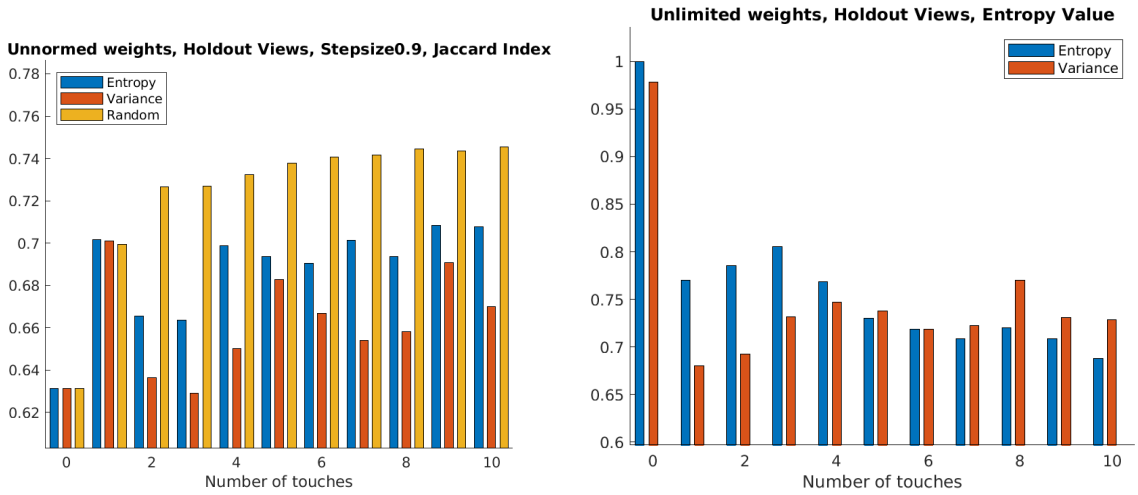


Figure A11: Unnormalized Weights, Holdout Views, Stepsize = 0.9

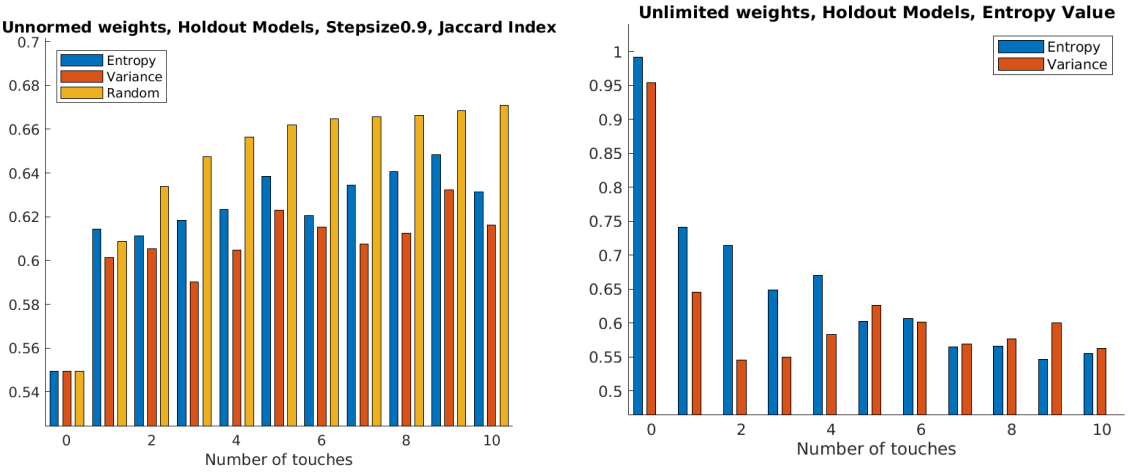


Figure A12: Unnormalized Weights, Holdout Models, Stepsize = 0.9

A.3 Stepsize = 0.8

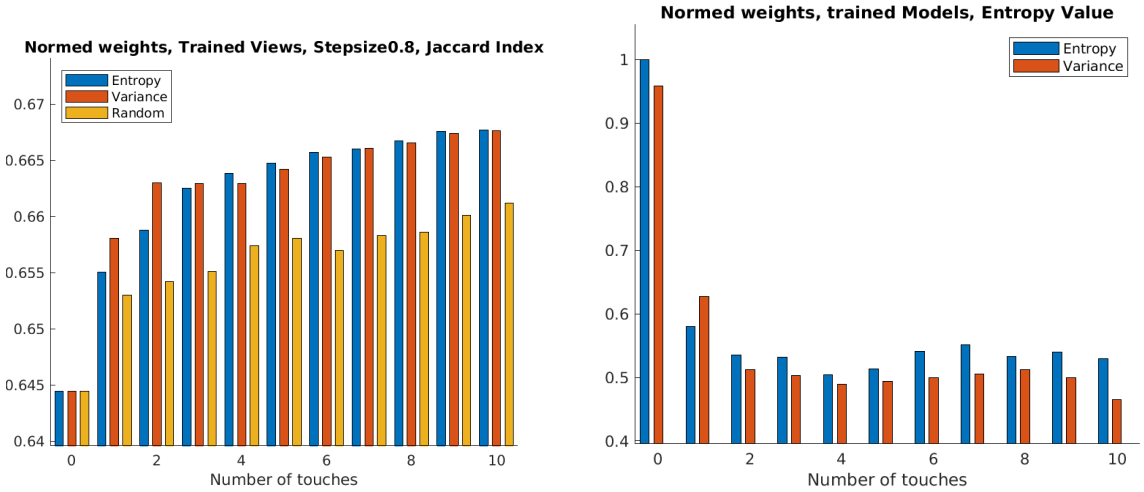


Figure A13: Normalized Weights, Trained Models, Stepsize = 0.8

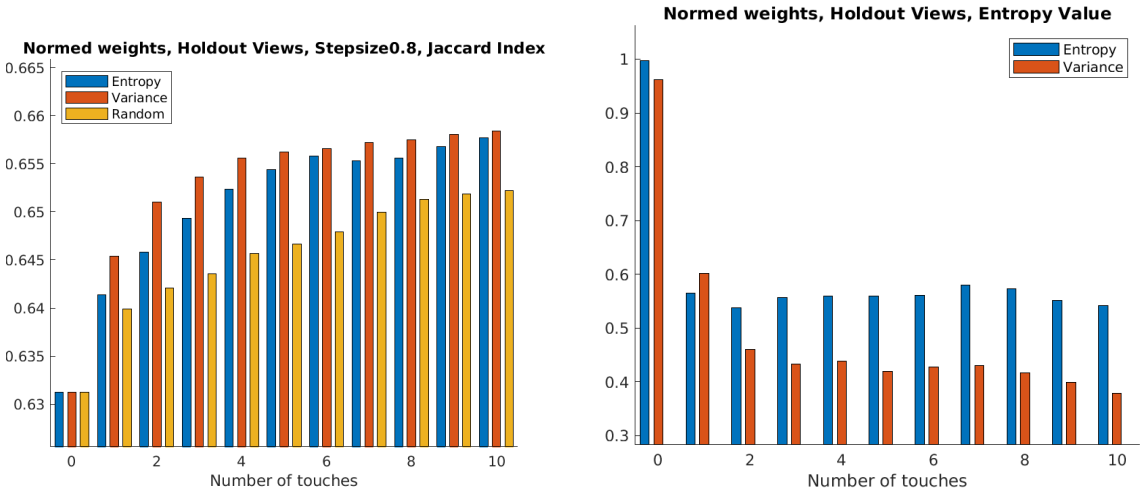


Figure A14: Normalized Weights, Holdout Views, Stepsize = 0.8

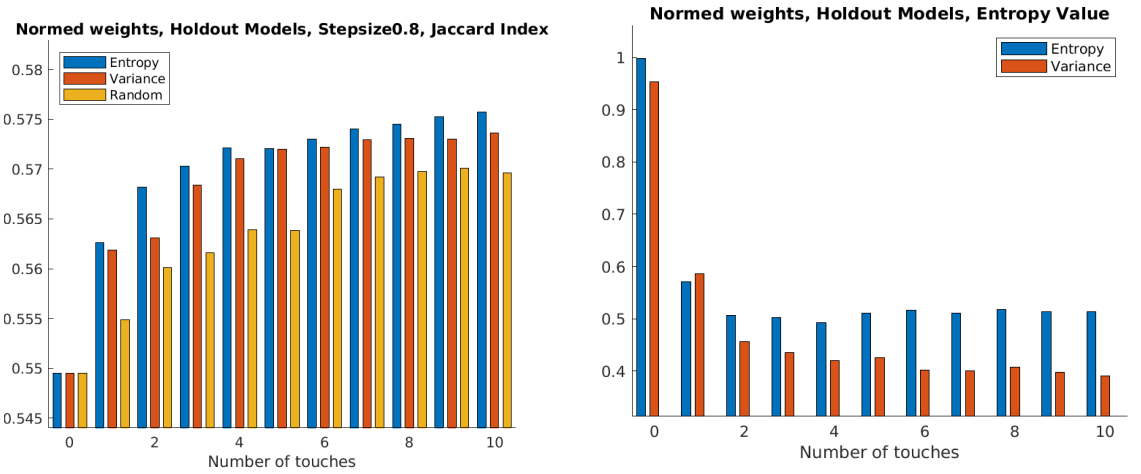


Figure A15: Normalized Weights, Holdout Models, Stepsize = 0.8

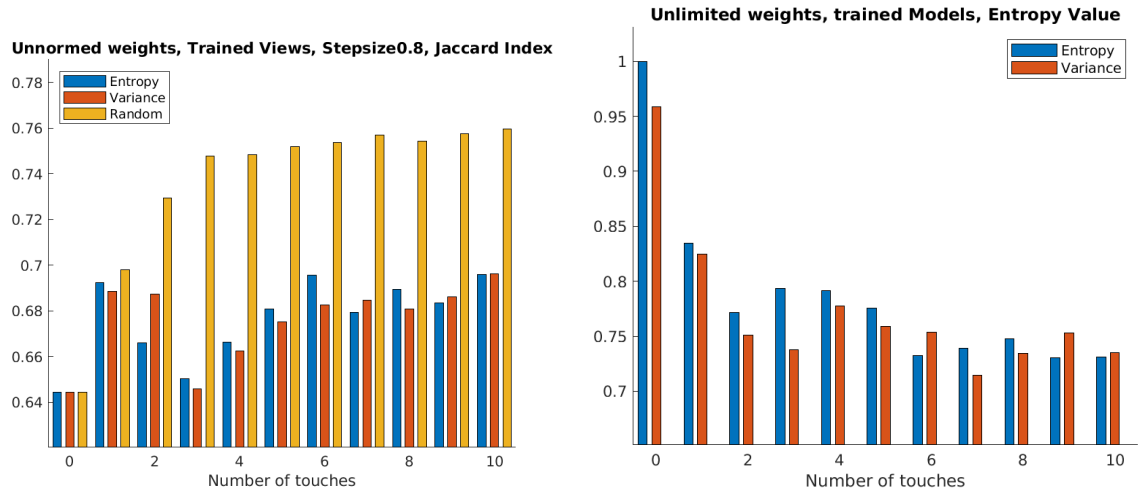


Figure A16: Unnormalized Weights, Trained Models, Stepsize = 0.8

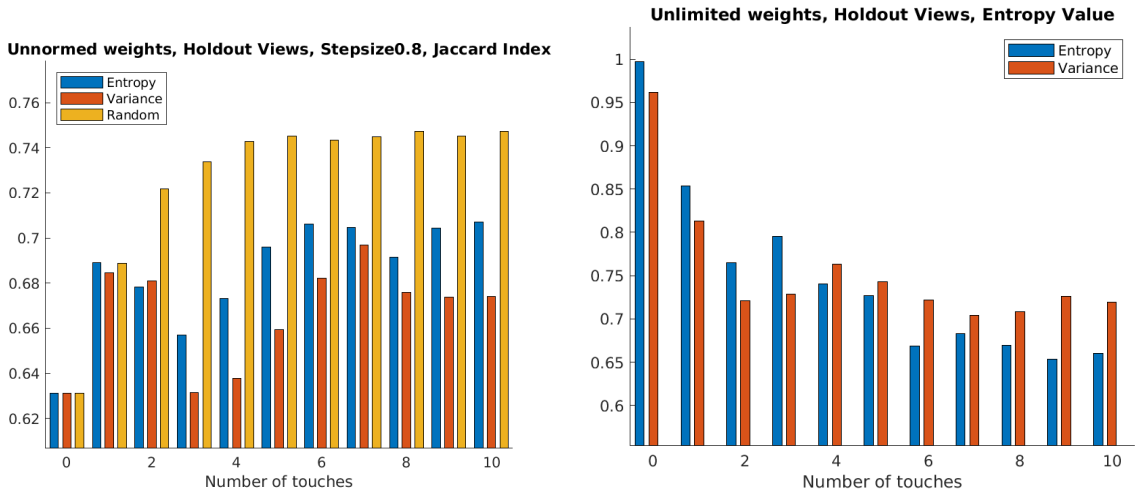


Figure A17: Unnormalized Weights, Holdout Views, Stepsize = 0.8

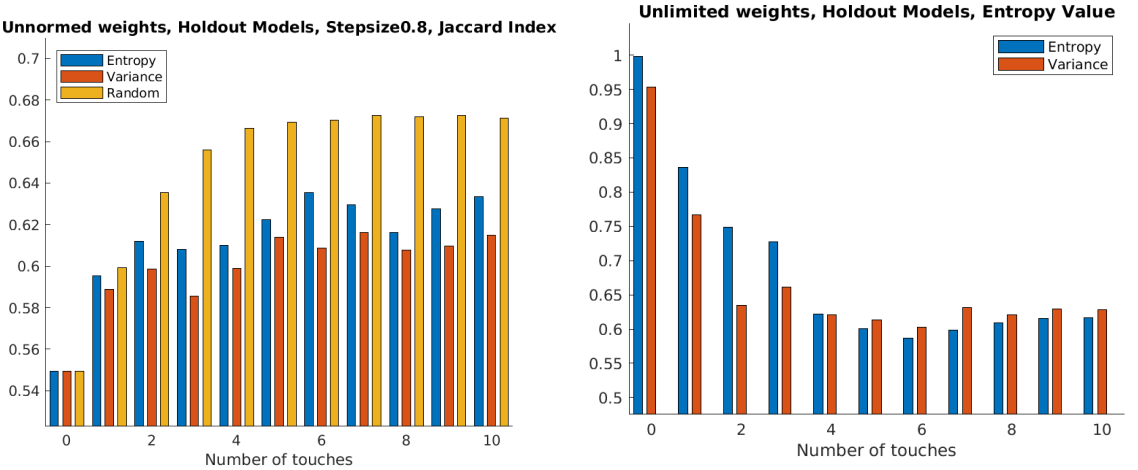


Figure A18: Unnormalized Weights, Holdout Models, Stepsize = 0.8

A.4 Stepsize = 0.6

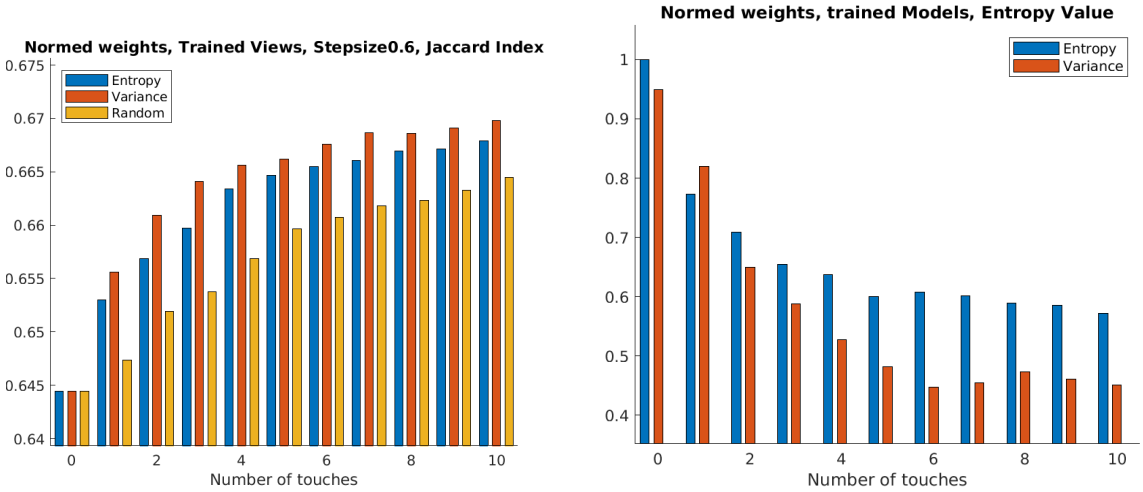


Figure A19: Normalized Weights, Trained Models, Stepsize = 0.6

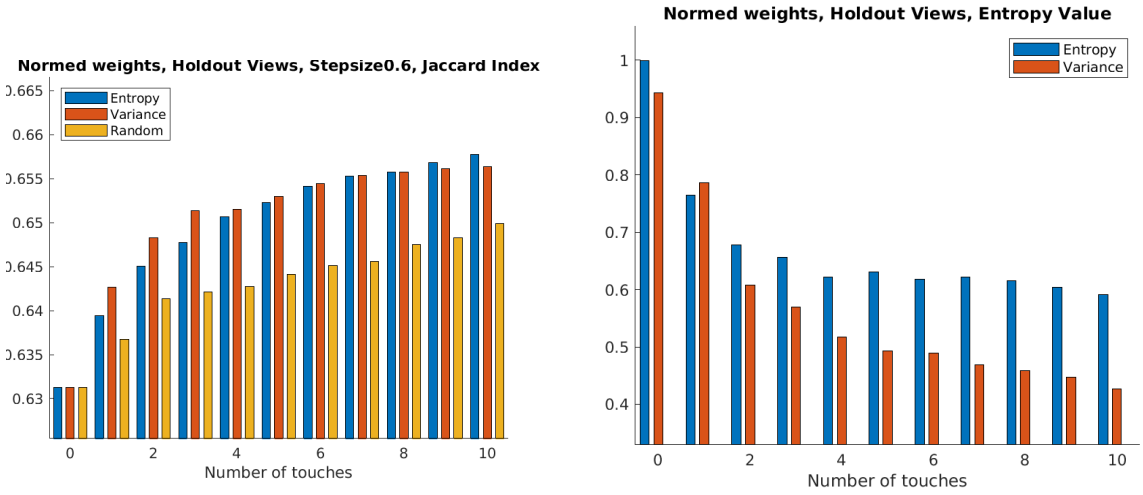


Figure A20: Normalized Weights, Holdout Views, Stepsize = 0.6

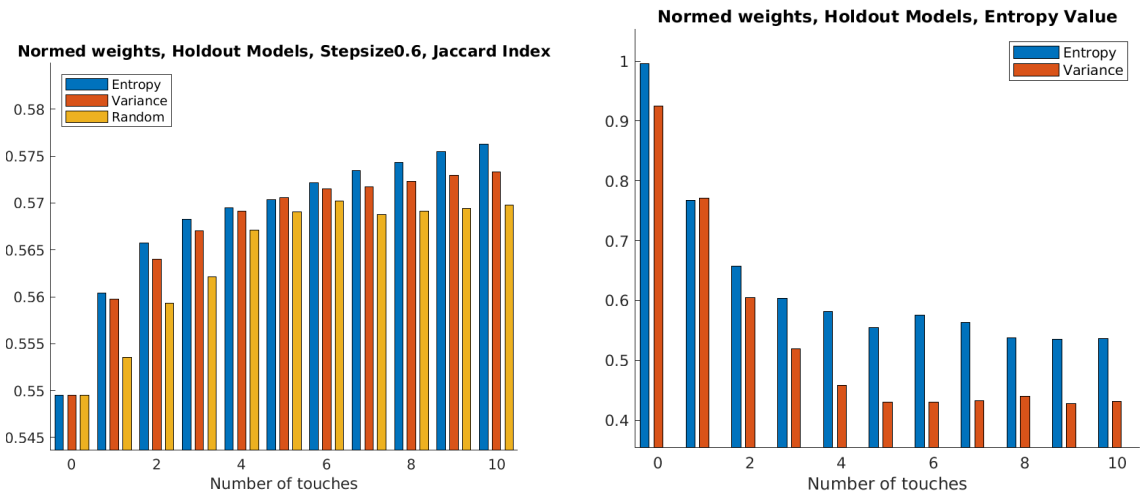


Figure A21: Normalized Weights, Holdout Models, Stepsize = 0.6

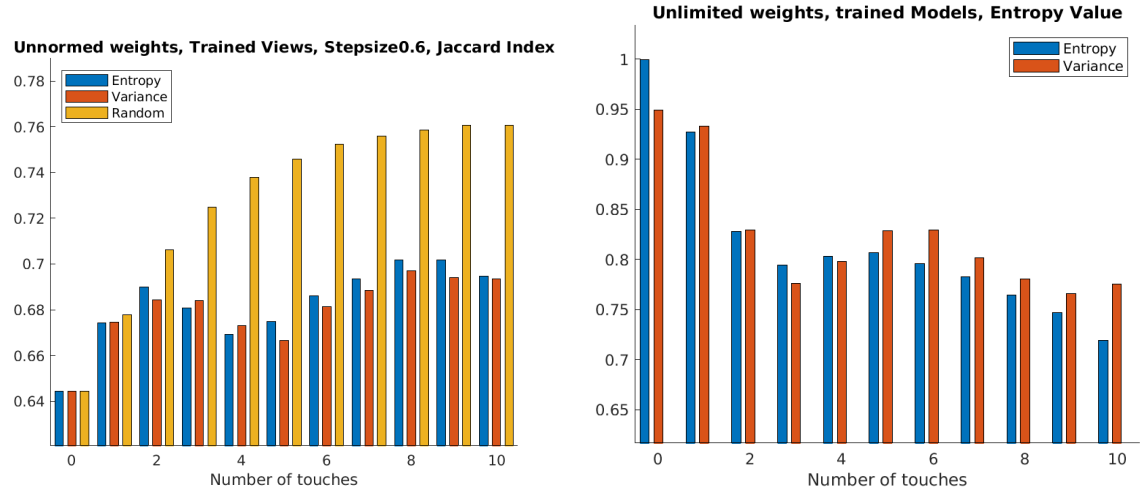


Figure A22: Unnormalized Weights, Trained Models, Stepsize = 0.6

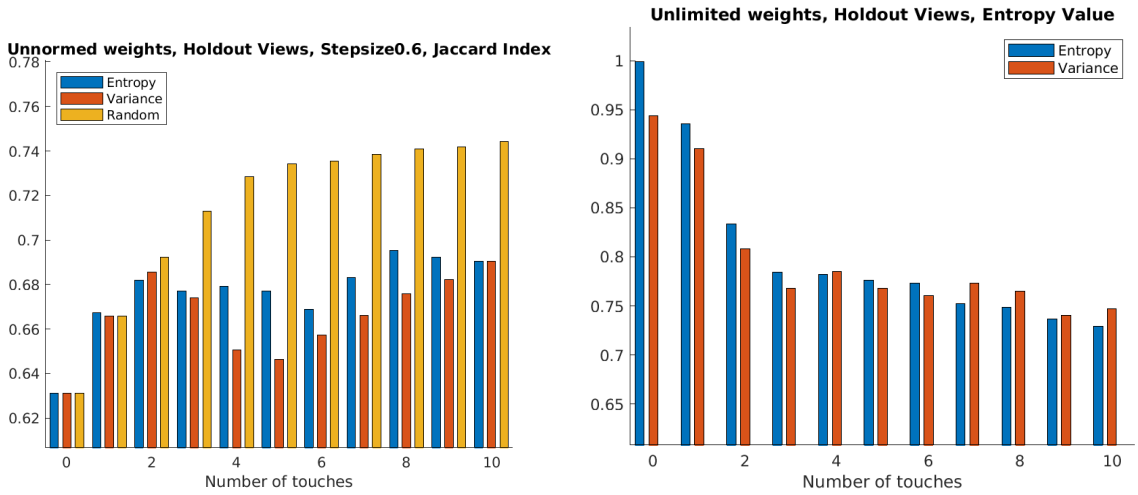


Figure A23: Unnormalized Weights, Holdout Views, Stepsize = 0.6

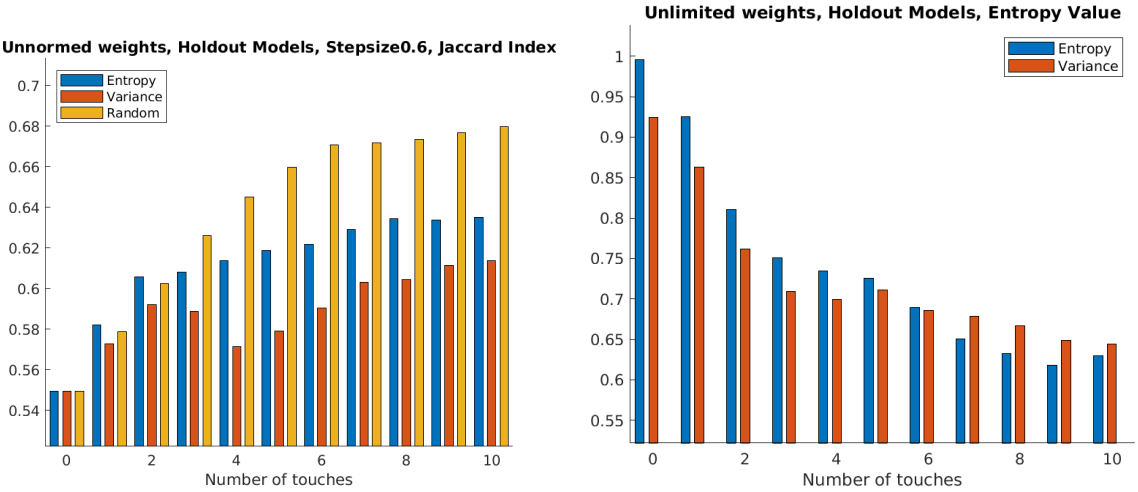


Figure A24: Unnormalized Weights, Holdout Models, Stepsize = 0.6

A.5 Stepsize = 0.4

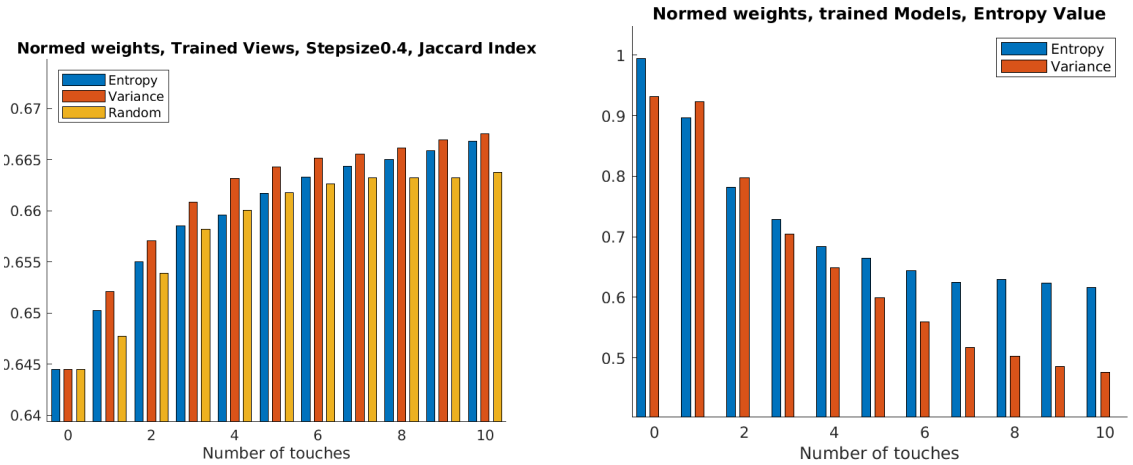


Figure A25: Normalized Weights, Trained Models, Stepsize = 0.4

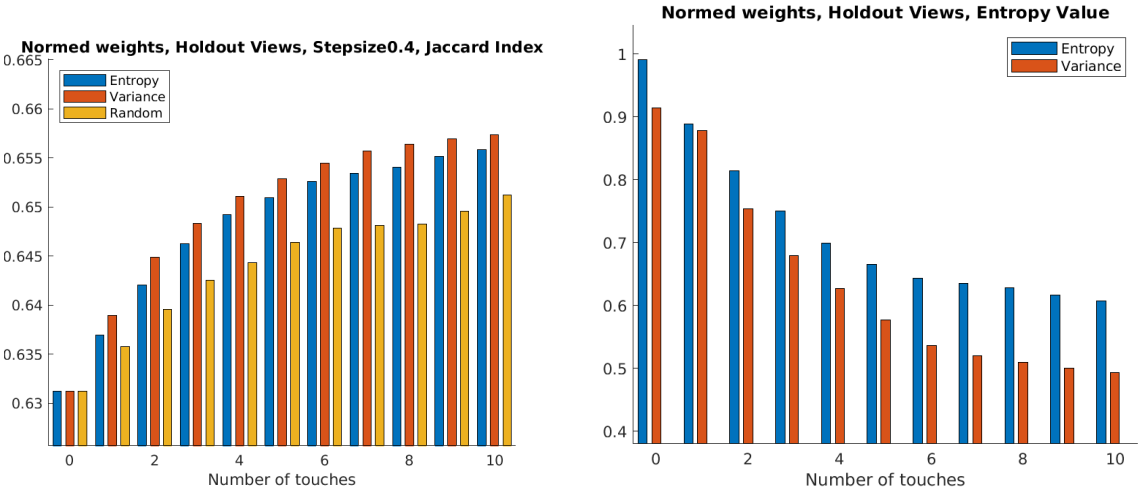


Figure A26: Normalized Weights, Holdout Views, Stepsize = 0.4

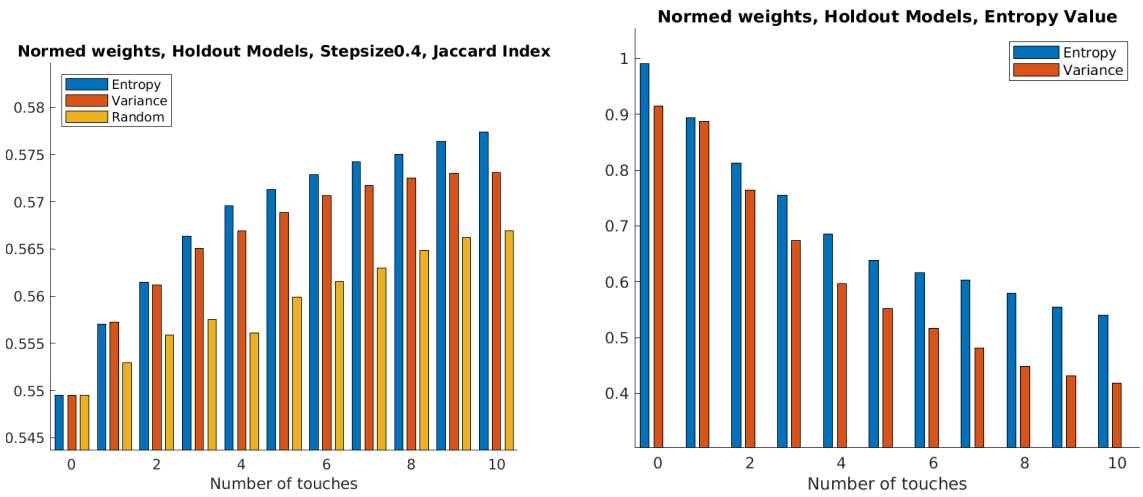


Figure A27: Normalized Weights, Holdout Models, Stepsize = 0.4

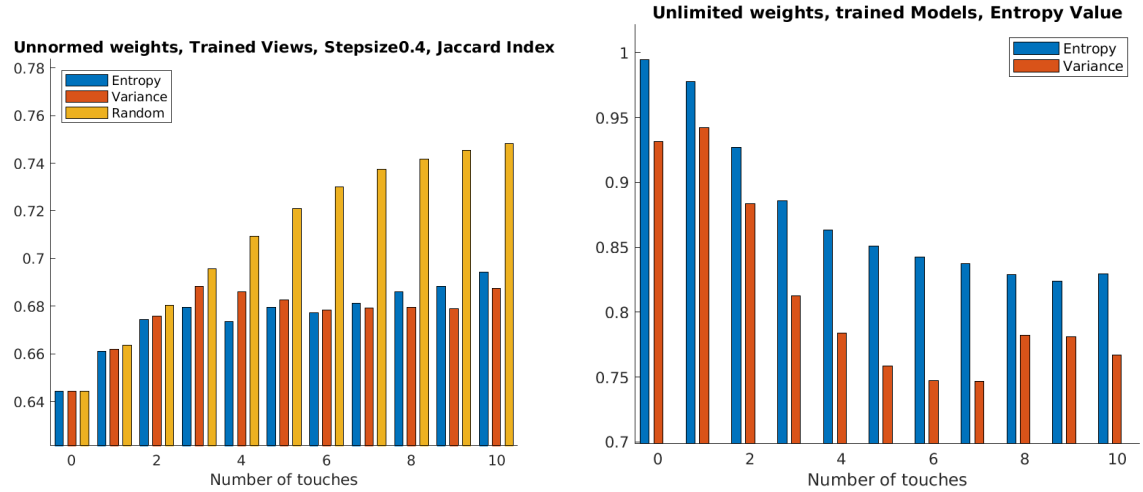


Figure A28: Unnormalized Weights, Trained Models, Stepsize = 0.4

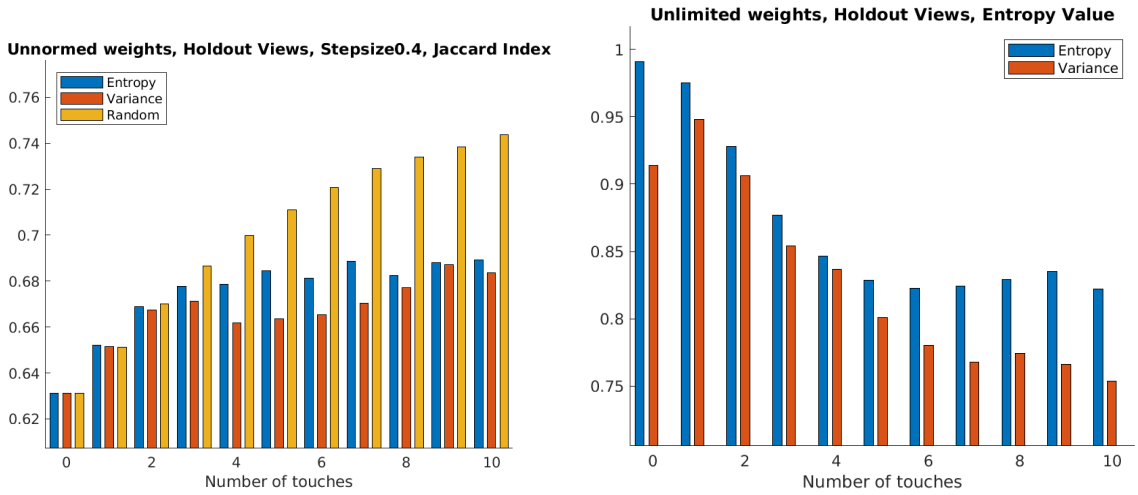


Figure A29: Unnormalized Weights, Holdout Views, Stepsize = 0.4

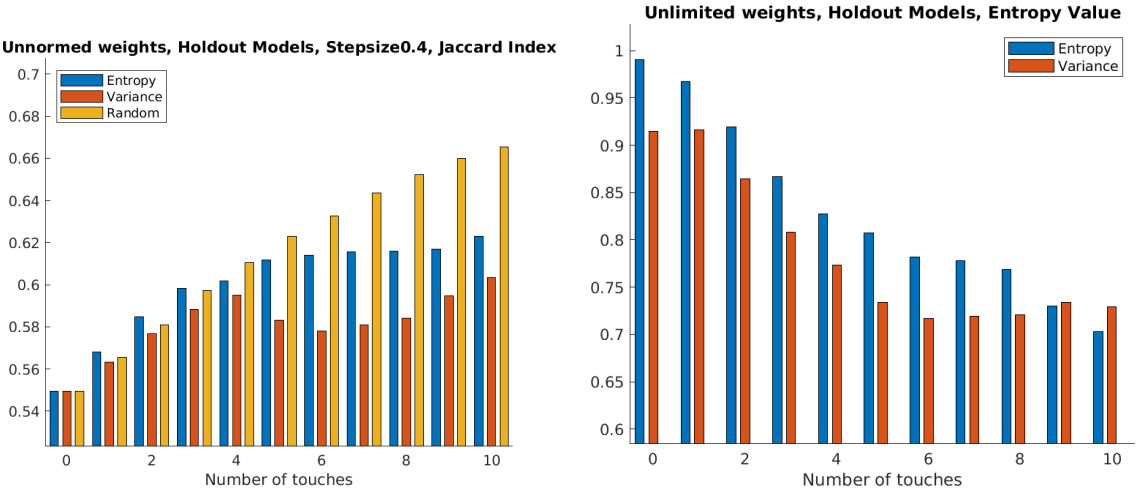


Figure A30: Unnormalized Weights, Holdout Models, Stepsize = 0.4